AVITRACK

Contract n° AST3-CT-3002-502818

Deliverable D3.1

Motion Detection

Version 1 – Draft 2

Project Number :	AST3-CT-3002-502818
Document number :	Deliverable D3.1
Document Title :	Motion Detection
Document status :	1.0 – Draft 2
Date :	22 September 2004
Availability :	Restricted
Authors :	UoR Team, PRIP Team, INRIA Team
Approved by:	

Abstract

This document describes the work on Motion Detection performed by the UoR team, the PRIP team and the INRIA team. Several motion detection algorithms were implemented and evaluated on a test video sequence acquired at Toulouse airport. The motion detection algorithms are compared with each other. The outcome of this work is the selection of one motion detection algorithm, or a combination of algorithms, for use in the AVITRACK project.

Keyword List

Motion Detection, Background Subtraction.

‘DOCUMENT CHANGE LOG

Document Issue.	Date	Reasons for change
Version 1 Draft 1	30-July-2004	Initial version.
Version 1 Draft 2	22-September-2004	Minor changes.

APPLICABLE AND REFERENCE DOCUMENTS (A/R)

A/R	Reference	Title

Table of contents

1. INTRODUCTION

This document describes the work on Motion Detection performed by the University of Reading (UoR) team, the PRIP team and the INRIA team. Several motion detection algorithms were implemented by each team and these algorithms were evaluated using a test video sequence acquired at Toulouse airport. The results of these motion detection algorithms are then evaluated against a set of criteria chosen to check their robustness and detection rates. The outcome of this work is the selection of one motion detection algorithm, or a combination of algorithms, for use in the AVITRACK project.

2. MOTION DETECTION – AN OVERVIEW

The output of a Motion Detector is a segmentation of an image into pixels labelled *background* and *foreground*, where the foreground pixels indicate the moving objects of interest.

Motion detection algorithms can be broadly classified into two types – those that use some form of *frame to frame differencing* to detect changes from one frame to the next; and those that use a *background model* to represent the scene, and detect motion by finding changes between the background model and the current frame. The latter are called *background subtraction* algorithms and are the most common type of motion detectors.

Normally, the first step of a background subtraction algorithm consists of learning the background model from an initial set of frames from the video sequence. Some of these algorithms assume that no significant foreground motion occurs during this period. Once the detection phase starts, this background model must be updated to reflect any short-term and long-term changes that occur in the background scene – this is called *background adaptation*. Without background adaptation, the background model starts to get out of date and the rate of detection errors increases.

Most motion detection algorithms work on a pixel-by-pixel level, but some use region level information, by considering connected regions of pixels, pixel neighbourhoods, etc. To handle wide variations in the environment, some algorithms use multiple background models and use a mechanism to switch from one model to another one that better explains the background scene at that particular time.

The output of a motion detector is normally used to find foreground regions (e.g. connected components), which can then be used by the tracking algorithms to track objects of interest across multiple frames. In some cases, the tracking algorithms can provide feedback to the motion detector to make the detection more robust.

3. AVITRACK WORK PACKAGE 3, TASK 3.1

For the AVITRACK project, Motion Detection is the first task of Work Package 3, Scene Tracking. The outcome of the Scene Tracking results depend heavily on the quality of motion detection, as errors from the motion detector module will propagate to and influence the scene tracking module, and later modules such as scene understanding.

The airport apron, being an outdoor environment, provides a number of challenges to motion detection. It must handle a wide range of environmental conditions, illumination changes and weather. The illumination changes can be long-term, such as the diurnal cycle, or short-term, caused by cloud movements, reflections, etc. The motion detector must be robust to such changes. Another constraint is that it must operate in real-time (at 12.5 frames per second, image size 768x576).

4. TEST VIDEO SEQUENCE

The motion detection algorithms developed for AVITRACK, were evaluated using a test video sequence, “Movie 1”, which was acquired at Toulouse Airport on 3 June 2004. Although this sequence was not acquired using the actual hardware installation that will be used at the airport (as this was still being installed at the time), it is still a good representative of what the input sequences will eventually look like. For example, it shows large short-term illumination changes, caused by cloud movement and bright sunlight, that can be expected in an airport scene.

The movie is stored as a sequence of individual JPEG files, with image size 768 by 576, and a rate of 12.5 frames per second. It is available for download from `ftp://ftp.cs.rdg.ac.uk/` (restricted access). The sequence consists of 4720 frames, giving a total duration of around 6 minutes. No major activity occurs before frame 712, so allowing the first 700 frames (≈ 56 seconds) to be used for background construction.

5. MOTION DETECTION ALGORITHMS

The following is the list of motion detection algorithms developed by the UoR, PRIP and INRIA teams [1–3]. These are described in more detail in the next sections.

1.	**Adjacent Frame Differencing**	PRIP team
2.	**Adjacent Frame Differencing and Morphology**	PRIP team
3.	**Mean and Threshold**	PRIP team
4.	**Mean and Standard Deviation**	PRIP team
5.	**W^4**	PRIP team
6.	**Frame Differencing and Gaussian**	PRIP team
7.	**Median and Morphology**	PRIP team
8.	**Kalman Filter**	PRIP team
9.	**Chromatic Background Model**	PRIP team
10.	**Median Filter**	UoR team
11.	**Colour Mean and Variance** • RGB • Normalised RGB + Chromatic tests • HSV + Chromatic tests	UoR team
12.	**Colour and Edge Fusion**	UoR team
13.	**Gaussian Mixture Model** • RGB • Normalised RGB + Chromatic tests	UoR team

14.	**Kernel Density Estimation**	UoR team
15.	**Linear Prediction (Wallflower)**	UoR team
16.	**Mean RGB**	INRIA team

6. DESCRIPTION OF THE ALGORITHMS

6.1. ADJACENT FRAME DIFFERENCING

Each frame is subtracted from the previous one. To reduce noise the image is convolved with a 5x5-Gaussian kernel. A threshold is set manually for the difference image. Due to the fact that only the previous frame is used as background model, only the outline of movement of homogenously coloured objects is detected. Furthermore, if an object stops moving, it is integrated in the background within one frame, which might yield to problems in person detection and tracking. Another problem of this method is the use of a global constant threshold that means that if illumination and contrast changes detection results might vary.

6.2. ADJACENT FRAME DIFFERENCING AND MORPHOLOGY

The difference image is computed in the same way as in Frame Differencing. To achieve a better result the difference image is subject to morphological opening and morphological closing in order to overcome the problem of detection motion in homogenously coloured objects. Slight improvements in comparison with adjacent frame differencing can be noticed, especially in detecting motion of small homogenous coloured objects; however this method faces the same problem concerning objects becoming motionless as frame differencing and concerning homogenous coloured objects bigger than a certain size (depending on the size of the closing structure element).

6.3. MEAN AND THRESHOLD

A weighted sum of input images is used to calculate the background model. Background subtraction is performed and the threshold is set manually. The resulting difference image is subject to morphological operations as described in Frame Differencing and Morphology. Due to the fact that the running average of the previous frames is used as background model, objects are not immediately integrated into background as it is done in adjacent frame differencing and detection of homogenous coloured objects improves. However, a global threshold has to be set manually.

6.4. MEAN AND STANDARD DEVIATION

The background model is established in the same way as Mean and Threshold. Additionally to the mean the standard deviation is calculated. The threshold is determined automatically by the standard deviation of each pixel. A pixel is considered foreground if it is out of the range of *k* standard deviations – that means, there is no need to set a global constant threshold.

6.5. W^4

W^4 was first presented by Haritaoglu et al. [4] in 1998; an improved algorithm was presented in 2000. In the experiments we implemented the background maintenance task of W^4. For each pixel a minimum and maximum intensity value is calculated as well as the maximum inter-frame intensity change. These parameters are regularly updated. If a pixel is outside of these parameters it is recognized as foreground. Morphological operations are performed to improve the result. The main advantage of this onset is that no global threshold needs to be set. However, especially in noised images, the algorithm yields insufficient results.

6.6. FRAME DIFFERENCING AND GAUSSIAN

Bramberger et al. [5] presents a Gaussian background model in connection with frame differencing to detect long-term background change. The background model pixels are initialized with a Gaussian. The mean and variance of the observed distribution are estimated according to the comparable ones of the background distribution calculated these during the last k frames. The background distribution is updated frame to frame and adopted with regard to the observed one.
The method is not robust against noise.

6.7. MEDIAN AND MORPHOLOGY

In this onset, the background model is computed as 50 median of consecutive images for all three colour channels. Thus, moved object can be better eliminated from the reference image. The reference image is updated according to a learning rate set by the user. This learning rate determines which frame should be used for calculation of the median. Each frame is subtracted from the background model and an adaptive threshold is used to detect motion pixels. The difference image is improved by morphological operations (opening to eliminate isolated pixels and closing to fill holes that might appear in homogenous coloured moving objects). Due to the use of an adaptive threshold, problems that are caused by a global constant threshold are compensated. A problem this onset faces is the adjustment of the learning rate. If the learning rate is too high (i. e. more frequent updates) a fading effect may occur, if it is too low, illumination changes are not adapted fast enough and therefore the majority of the pixels may be detected as foreground. Further problems occur in detection of slow moving homogenous coloured objects as they may be integrated into the background too fast.

6.8. KALMAN FILTER

The Kalman Filter is a prediction filter based on the assumption that the best information of a system state is obtained by an estimation, which explicitly considers noise. In [6, 7] the possible use of Kalman Filters for background/foreground segmentation is described. For each frame t_i a prediction $x_p(t_i)$ is calculated, which is compared with the actual measured value, and a new system estimation $x_e(t_i)$ is obtained by weighting the difference of prediction and measured value. This is done in the following way.

$$x_p(t_i) = A(t_i) \cdot x_e(t_{i-1})$$

and

$$x_e(t_i) = x_p(t_i) + K(t_i) \cdot [z(t_i) - H(t_i) \cdot x_p(t_i)]$$

where $A(t_i)$ is called system matrix, $H(t_i)$ a measurement matrix, $z(t_i)$ is the system input (the actual pixel value), and $K(t_i)$ is called Kalman gain matrix, which is derived from the error covariance matrix that means the gain is high if the measurement noise is low. In the current implementation A and H are constant for each pixel in each frame. The Kalman gain for a pixel in frame t_i is set to α if this pixel was detected as foreground in frame t_{i-1}, otherwise it is set to β. To segment foreground and background, a difference image of the estimated image and the actual image is calculated. An adaptive threshold is applied on the difference image. This thresholded image is subject to morphological opening and closing. One problem that appears in using Kalman filters is a certain amount of fading.

6.9. CHROMATIC BACKGROUND MODEL

Horprasert et al. [8] proposed a chromatic background model able to cope with illumination changes such as shadows and highlights. The background model is updated frame to frame by a Kalman Filter. Brightness Distortion (ϕ) and Colour Distortion (CD) are calculated for each pixel. They are obtained by minimizing

$$\phi(\alpha_i) = (I_i - \alpha_i E_i)^2$$

and

$$CD_i = \| I_i - \alpha_i E_i \|$$

Let i be a pixel in the image; $I_i = [I_R(i), I_G(i), I_B(i)]$ represents the pixel's RGB colour value in a current image, $E_i = [E_R(i), E_G(i), E_B(i)]$ the pixel's expected RGB colour in the background model and α_i the pixel's strength of brightness with respect to the expected value.

A pixel is classified as foreground if CD_i is greater than a threshold or $\phi(\alpha_i)$ is less than another one, as background if $\phi(\alpha_i)$ is close to 1 and as shadow if $\phi(\alpha_i)$ is less than 1 (thresholded). Otherwise it is considered as highlighted pixel. Thresholds are determined by a statistically learning procedure. The resulting foreground image is subject to morphological opening and closing.

The method suffers the problem of reflection (mirror, water…) where pixels are classified as false positives. Another problem arises in front of white/grey targets walking against a grey background. In this case there is no colour information and the target will be not detected.

6.10. MEDIAN FILTER METHOD

This motion detector was developed for the *Reading People Tracker* program, by Nils T. Siebel [9]. It operates on greyscale images, so the video images are pre-processed to convert from RGB to greyscale using the CIE Y_{709} conversion function:

$$x = Y_{709}(R,G,B) = 0.2125R + 0.7154G + 0.0721B$$

This motion detector uses the median value *m* per pixel for the background representation. A pixel is marked as "foreground" if the absolute difference between the median *m* and the pixel value *x* is larger than a global threshold DETECT_DIFF_THRESHOLD (chosen manually):

$$| x - m | > \text{DETECT\_DIFF\_THRESHOLD}$$

The output image is then post-processed to fill small holes by performing a number of morphological dilations, specified by parameter GAP_SIZE, and using a 3x3 structuring element.

Initially, the background is set to be the first frame in the sequence. The background model is then updated by looking for runs of monotonically increasing or monotonically decreasing pixel values. When such a run of values is found for a particular pixel, the median *m* for that pixel is replaced with the last value of the run. The run length is specified by the global parameter BACKGROUND_RUN_LENGTH.

$m_{t+1} = x_t$ if $x_t >= x_{t-1} >= x_{t-2} >= \ldots >= x_{t-R+2} >= x_{t-R+1} >= x_{t-R}$
or $x_t <= x_{t-1} <= x_{t-2} <= \ldots <= x_{t-R+2} <= x_{t-R+1} <= x_{t-R}$
where R is the BACKGROUND_RUN_LENGTH.

This method of background adaptation allows for a faster generation of the median value. The rationale in only updating the median if there was a monotonic run, is that outdoor lighting changes usually create a monotonically increasing or decreasing intensity values, so ignoring alternating intensity changes, probably due to noise or foreground motion.

This motion detector is simple and runs very fast. Its main disadvantage is that it requires the difference threshold to be selected manually. When updating the background, no distinction is made between pixels labelled as background or foreground. So, the adaptation of this detector to illumination changes and integration of foreground objects into the background depend on the same parameter, the BACKGROUND_RUN_LENGTH. When a low value was used, foreground objects were integrated very quickly, resulting in only the outline being detected for slow-moving objects, as well as leaving trailing ghosts. Using a high value, the detector did not adapt fast enough to illumination changes, giving rise to a lot of false positives.

6.11. COLOUR MEAN AND VARIANCE METHOD

6.11.1. RGB COLOUR SPACE

The background model of this motion detection algorithm is represented by a pixel-wise mean value μ and variance σ^2. This is based on the assumption that a background pixel can be modelled with a Gaussian distribution $N(\mu,\sigma^2)$ to handle the image noise. The first version of this algorithm to be implemented uses the standard RGB colour space, so the background model B for each pixel is:

$$B = (\mu, \sigma) = (\mu = \langle \mu_R, \mu_G, \mu_B \rangle, \sigma = \langle \sigma_R, \sigma_G, \sigma_B \rangle)$$

Motion is detected by taking the absolute differences between the pixel value x and the mean μ for each colour channel. If the difference for any of the colour channels is larger than 3.5σ, then the pixel is marked as “foreground”. While if less than 2.5σ, then the pixel is marked as “background”. If the differences are in between 2.5σ and 3.5σ, then the pixel is marked as “probable” foreground pixel; these will then be examined later during post-processing to classify them into “foreground” or “background” based on the evidence from their neighbouring pixels.

$$\Delta_C = | x_C - \mu_C | \qquad \text{for } c \in \{R,G,B\}$$

If	$\Delta_R > 3.5\sigma_R$ or $\Delta_G > 3.5\sigma_G$ or $\Delta_B > 3.5\sigma_B$	pixel is foreground
If	$\Delta_R > 2.5\sigma_R$ or $\Delta_G > 2.5\sigma_G$ or $\Delta_B > 2.5\sigma_B$	pixel is “probable” foreground
Else		pixel is background

In contrast to the previous algorithm, the threshold is not fixed for the whole image. The choice of the 3.5σ limit represents a 99.9% probability of the value not belonging to the Gaussian distribution $N(\mu,\sigma^2)$, while 2.5σ is a reasonable lower limit.

Motion detection is followed by thresholding by hysterisis to classify the “probable” foreground pixels, based on the evidence from its 8 neighbours, as well as filling holes and suppressing noise pixels. The rules used are:

- If a pixel is marked as “probable” and more then half its neighbours are marked as “foreground”, then it is promoted to “foreground”.
- If a pixel is marked as “foreground” and less than half of its neighbours are marked as “foreground”, then it is demoted to “probable”.
- If a pixel is marked as “background” and all its neighbours are marked as “foreground”, then it is promoted to “foreground”.

The number of iterations used during thresholding by hyserisis is controlled by the parameter HYSTERESIS_THRESHOLDING_ITERATIONS. After the last iteration, any remaining pixels marked as "probable" are labelled as "background" pixels.

The background model is learnt from the first N frames of the image sequence, where N is specified by parameter BACKGROUND_RUN_LENGTH:

$$\mu_C = \frac{1}{N}\sum_{t=1}^{N} x_c(t) \qquad c \in \{R,G,B\}$$

$$\sigma_C = \sqrt{\frac{1}{N}\sum_{t=1}^{N} x_c(t) - \mu_c^{\,2}}$$

The model is then updated to handle changes in illumination by incorporating part of the new pixel values, determined by the learning rate. Two different learning rates are used – one for pixels marked as "background" and the other for those marked as "foreground", specified by parameters BACKGROUND_UPDATE_WEIGHT and FOREGROUND_UPDATE_WEIGHT.

$$\mu_C(t+1) = (1-\alpha)\,\mu_C(t) + \alpha\, x_C(t)$$

$$\sigma_C(t+1) = (1-\alpha)\,\sigma_C(t) + \alpha\,|\, x_C(t) - \mu_C(t)\,|$$

where α = BACKGROUND_UPDATE_WEIGHT, if $x(t)$ is marked as "background" pixel
α = FOREGROUND_UPDATE_WEIGHT, otherwise.

When updating the standard deviation σ, an absolute difference is used instead of the more accurate square-root function, for performance reasons. A check is also performed to ensure that the value of σ_C never becomes 0 after the background update. If it were allowed to be 0, then that particular pixel will be erroneously marked as "foreground" in subsequent frames. If σ_C becomes 0, then it is set to a minimum value specified by the global parameter MIN_SIGMA.

This motion detector runs quite fast, and does not depend on a global threshold when compared to the previous one. When tested on movie sequence 1, some false positives occurred caused by sudden illumination changes that happen at a much faster rate than the rate of background update. Another problem is caused by foreground objects being too similar to the background. This causes holes to appear in foreground objects and object fragmentation.

6.11.2. NORMALISED RGB COLOUR SPACE & CHROMATIC TESTS

This motion detection algorithm is based on the algorithm described in the previous section (§6.11.1). The Normalised RGB colour space is used instead – this colour space separates the chromaticity component of the colour from the brightness component. The algorithm was extended to take advantage of this chromaticity/brightness separation to make motion detection more illumination invariant.

$$(R,G,B) \rightarrow (r,g,b,s): \quad r = R/(R+G+B),\; g = G/(R+G+B),\; b = B/(R+G+B),\; s = (R+G+B)/3$$

The background model B is defined in terms of both the chromaticity components (r,g,b) as well as the brightness component (s):

$$B = (\mu, \sigma) = (\mu = \langle \mu_r, \mu_g, \mu_b, \mu_s \rangle, \sigma = \langle \sigma_r, \sigma_g, \sigma_b, \sigma_s \rangle)$$

Motion detection happens in the same way as described for the algorithm in the previous section, with the difference that the pixel classification is extended to include "shaded background" and "highlighted background" in addition to "background", "foreground", and "probable". The conditions for marking pixels as "shaded background" and "highlighted background" is based in part on the work of Horprasert et al [8]. This is based on the idea that if a pixel is not different in the chromaticity components (r,g,b) from its background model, but there is a difference in the brightness component (s), then the change is most probably due to shadow or highlight (e.g. specular reflections).

$\Delta_S = | g_s - \mu_s |$

If *no chromaticity difference*:

If $g_S < \mu_S$ and $\Delta_S > \tau_{\alpha 1} \mu_S$ $(\tau_{\alpha 1} <= 1)$ pixel is "shaded background"

If $g_S > \mu_S$ and $\Delta_S < \tau_{\alpha 2} \mu_S$ $(\tau_{\alpha 2} >= 1)$ pixel is "highlighted background"

where the global constants $\tau_{\alpha 1}$ and $\tau_{\alpha 2}$ control the maximum amount of darkening or brightening in the scene. In [8], it is mentioned that these thresholds can be determined by a statistical learning procedure. In the current implementation of the algorithm, these two constants are specified manually through the parameters MAX_SHADOW_DARKENING and MAX_HIGHLIGHT_BRIGHTENING. This algorithm can be changed in the future to determine these parameters automatically.

In the thresholding by hysterisis step, extra rules are used to handle pixels marked as "shaded background" and "highlighted background":

- If a pixel is marked as "probable" and more then half its neighbours are marked as "shaded background", then it is promoted to "shaded background".
- If a pixel is marked as "probable" and more then half its neighbours are marked as "highlighted background", then it is promoted to "highlighted background".
- If a pixel is marked as "shaded background" and less than half of its neighbours are marked as "shaded background", then it is demoted to "probable".
- If a pixel is marked as "highlighted background" and less than half of its neighbours are marked as "highlighted background", then it is demoted to "probable".

The process for learning the initial background model and the process for updating the model, are the same as in the previous section.

Unlike the previous one, this motion detector is able to handle correctly sudden illumination changes caused by the sun going behind clouds, giving very few false positives. The disadvantage is that two new parameters were introduced. At the moment, these are configured manually. But this can be changed in the future by implementing the method mentioned by [8] to set these parameters automatically. Another problem is caused by strong shadows cast by foreground objects on parts of the apron that contains no colour information – these are detected as false positives.

6.11.3. HSV COLOUR SPACE & CHROMATIC TESTS

A third variation of the Colour Mean and Variance algorithm was implemented using the HSV colour space. Similarly to the case of Normalised RGB, the HSV colour space separates the chromaticity and brightness components, so allowing the identification of "shaded background" and "highlighted background". The background model B is:

$$B = (\mu, \sigma) = (\mu = <\mu_H, \mu_S, \mu_V>, \sigma = <\sigma_H, \sigma_S, \sigma_V>)$$

Unlike the normalised RGB space, HSV can have an unstable hue H for a very low saturation S or a very low value V. These are eliminated by considering only those HSV colours with S > LOW_SV_THRESHOLD and V > LOW_SV_THRESHOLD. This parameter is set to 0.18.

During motion detection, if a pixel does not have a stable hue (is colour-less) or the background model's hue is not stable, then only the brightness information is used for detecting motion.

Likewise, when creating the initial background model from the first BACKGROUND_RUN_LENGTH images, only stable hue values are used, and an array is used to keep track of the number of stable hue samples used. This array is then used when calculating the initial μ, σ for that pixel.

Like the previous algorithm, this motion detector handles illumination changes quite well, but with the same drawback of having two new parameters to configure. When comparing the use of HSV versus Normalised RGB on movie 1, it was found that using HSV gives more errors than Normalised RGB due to JPEG noise.

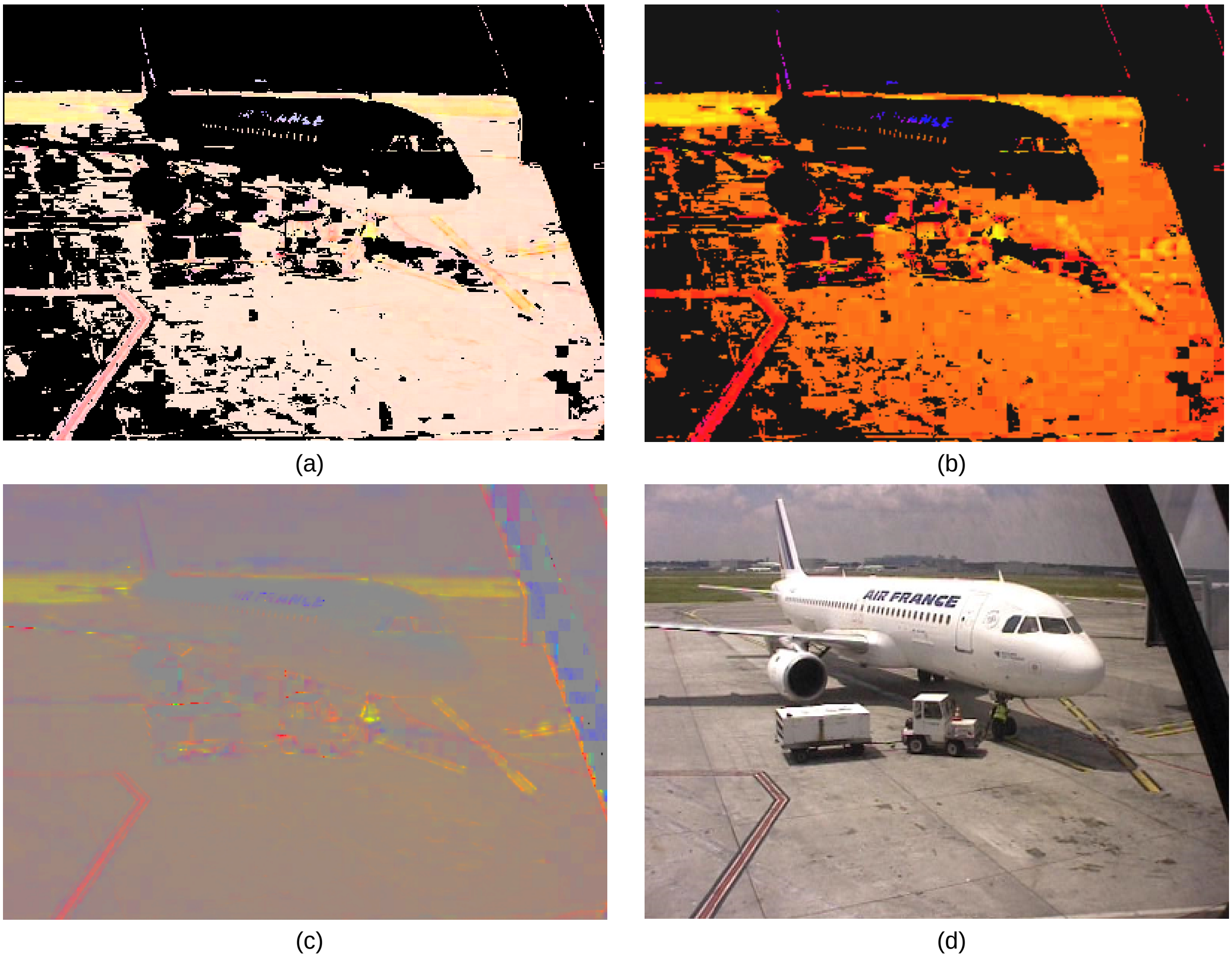


Figure 1: (a) Hue and Saturation – constant Value. (b) Hue only – constant Saturation and Value. Black pixels indicate areas of unstable Hue, and so are considered to be colour-less by the motion detector. (c) Normalised RGB. (d) The original RGB frame (#1656). Most of the foreground objects are either colour-less or have the same colour as the background.

6.12. COLOUR AND EDGE FUSION METHOD

This motion detection algorithm is based on the work of Jabri et al [10]. Edge information is combined together with colour and illumination to model the background.

The colour-based part of this motion detector was derived from the algorithm described in section 6.11.2, the Colour Mean and Variance method with Normalised RGB colour space.

The edge model information consists of the horizontal and vertical gradients calculated using 3x3 Sobel operators *S*x and *S*y on the greyscale image (the *s* component from Normalised RGB).

Sx=

-1	-2	-1
0	0	0
1	2	1

Sy=

-1	0	1
-2	0	2
-1	0	1

The background model B is defined as:

$$B = (\mu_{colour}, \sigma_{colour}, \mu_{edge}, \sigma_{edge}) = (\mu_{colour}, \sigma_{colour}, \mu_{edge} = \langle\mu_{Gx}, \mu_{Gy}\rangle, \sigma_{edge} = \langle\sigma_{Gx}, \sigma_{Gy}\rangle)$$

The colour-based part of motion detection happens exactly in the same way as described in section 4.3. This marks each pixel as either "foreground", "background", "shadowed background", "highlighted background", or "probable".

The edge-based part of motion detection compares the strength and the direction of the x and y gradients at each pixel *x* of the image with that of the background model.

$$\Delta_{Gx} = |\, g_{Gx} - \mu_{Gx} \,| \qquad \Delta_{Gy} = |\, g_{Gy} - \mu_{Gy} \,|$$

$$D_x = \begin{cases} 1 & \text{if } \mathrm{sign}(g_{Gx}) = \mathrm{sign}(\mu_{Gx}) \\ 0 & \text{if } \mathrm{sign}(g_{Gx}) \neq \mathrm{sign}(\mu_{Gx}) \end{cases} \qquad D_y = \begin{cases} 1 & \text{if } \mathrm{sign}(g_{Gy}) = \mathrm{sign}(\mu_{Gy}) \\ 0 & \text{if } \mathrm{sign}(g_{Gy}) \neq \mathrm{sign}(\mu_{Gy}) \end{cases}$$

If there is a significant edge in the background model and also in the image and both have the same direction, then the pixel is marked as "background edge". If they have different directions, then the pixel is marked as "occluded edge". If there is no significant edge in the background model, but there is a significant edge in the current image, then the pixel is marked as "new edge".

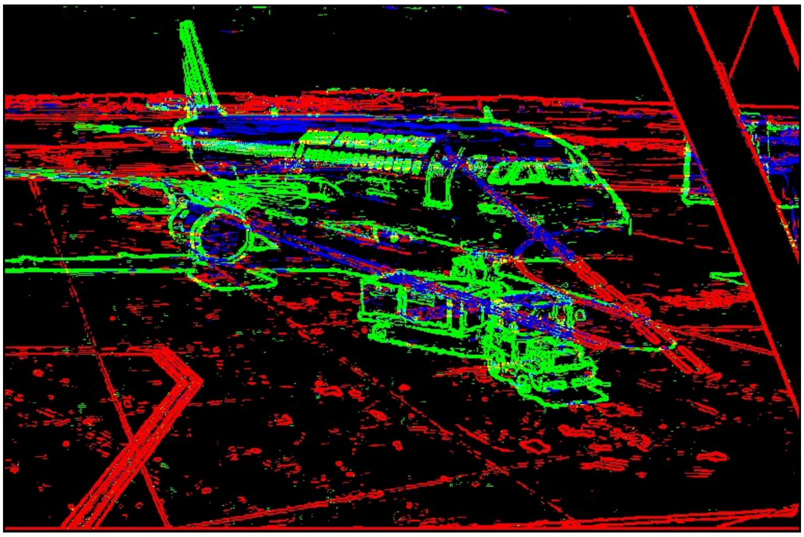


Figure 2: Background edges are red; Occluded background edges are blue; New edges are green. (Frame #1700)

The results of both the colour- and the edge-based motion detection are then combined together to arrive at a final labelling for all the pixels using the rules given in the table below. Pixels marked as "probable", for which there is not enough confidence in the label when using the colour information alone, the edge information is used to boost their confidence and re-label them as "foreground" or "background". In case of a conflict between the edge-based label and the colour-based label, then the pixel is set to "probable" and its classification is done later at the post-processing stage, by looking at its neighbouring pixels.

Edge-based label	Colour-based label	New label
"background edge"	"probable"	"background"
"background edge"	"foreground"	"probable"
"new edge"	"probable"	"foreground"
"new edge"	"background"	"probable"
"occluded edge"	"probable"	"foreground"
"occluded edge"	"background"	"probable"

Motion detection is followed by thresholding by hysterisis as described in section 6.11.2 to suppress noise and handle the remaining undecided pixels.

For the initial background model, the values μ_{edge} and σ_{edge} are derived in a similar way to that of the colour information. A parameter MIN_EDGE_SIGMA is defined to ensure that σ_{edge} is never 0.

At the moment, as implemented, this motion detection algorithm is not updating the edge information, when updating the background colour information. This is a limitation that may cause the edge information present in the background scene to become out-dated. But it can be easily included in the future.

The main advantage of this motion detector compared to the previous ones, is that by using edge information together with colour, the problem of holes in foreground objects and object fragmentation is reduced. The main disadvantage is a slower speed.

6.13. GAUSSIAN MIXTURE MODEL (GMM) METHOD

6.13.1. RGB COLOUR SPACE

The implementation of this algorithm is based on the work of Stauffer et al [11]. This motion detection algorithm is based on the idea that normally pixels of a background scene may represent different objects at different times; for example, a tree swaying in the wind. These can best be modelled using a number *K* of Gaussian trivariate distributions *N*, expressed as a mixture model, per pixel. The background model B is then:

$$P(x) = \sum_{t=1}^{K} \omega_i N(\mu, \Sigma)$$

$$B = \Big(\omega_1, \ldots, \omega_K, \\ \mu_1 = <\mu_{(1,R)}, \mu_{(1,G)}, \mu_{(1,B)}>, \ldots, \mu_K = <\mu_{(K,R)}, \mu_{(K,G)}, \mu_{(K,B)}>, \\ \Sigma_1 = \mathbf{I}<\sigma_{(1,R)}^2, \sigma_{(1,G)}^2, \sigma_{(1,B)}^2>, \ldots, \Sigma_K = \mathbf{I}<\sigma_{(K,R)}^2, \sigma_{(K,G)}^2, \sigma_{(K,B)}^2> \Big)$$

For both processing and storage efficiency reasons, the covariance matrix Σ is assumed to consist of the diagonal entries only; meaning that the colour components are assumed to be independent.

The number *K* of distributions is defined by parameter NUM_GMM_DISTRIBUTIONS, and was set to 4 for this implementation.

This motion detection algorithm allows some of the *K* distributions to be used to represent foreground objects, making the process of incorporating new objects into the background much easier. For each pixel *x*, the *K* distributions are stored in the mixture model in descending order of their weight ω_i. The first *J* distributions of these with a combined weight greater than threshold MAX_BACKGROUND_WEIGHT are considered to represent the background scene. The rest of the *K*-*J* distributions represent new objects that have been incorporated into the background model.

$$J = \underset{j}{\operatorname{argmin}}\left(\sum_{k=1}^{j} \omega_k > \text{MAX\_BACKGROUND\_WEIGHT} \right) \qquad (\omega_{k-1} >= \omega_k)$$

If a pixel x is within 3.5σ of any of the first J distributions, then it can be explained by that particular distribution and so is marked as "background". If the pixel is explained by any of the remaining K-J distributions, or is not explained by any of the distributions at all, it is then marked as "foreground". The value 3.5 for the number of sigmas, is configured through parameter NUM_SIGMA_THRESHOLD.

The resulting motion image is post-processed to remove isolated foreground pixels and to fill small holes in foreground regions.

The background model is updated as follows. If a pixel x matches any of the K distributions, then that distribution is updated and the rest are left unchanged. The distribution i is updated according to the following rules:

$$\omega_i(t+1) = (1-\alpha)\,\omega_i(t) + \alpha$$

$$\mu_{i,C}(t+1) = (1-\alpha)\,\mu_{i,C}(t) + \alpha\, x_C(t)$$

$$\sigma_{i,C}^2(t+1) = (1-\alpha)\,\sigma_{i,C}^2(t) + \alpha\,\left| x_C(t) - \mu_{i,C}(t) \right|^2$$

where α = BACKGROUND_UPDATE_WEIGHT, if x(t) is marked as "background" pixel
α = FOREGROUND_UPDATE_WEIGHT, otherwise.

If a pixel does not match any of the K distributions, then the one with the least weight ω is replaced with a new distribution defined as $N(\mu=x, \Sigma)$, with an initially large variance Σ and an initially very small weight ω.

The Expectation-Maximisation (EM) algorithm is used to learn the initial background model from the first number of frames of the video sequence. Parameter BACKGROUND_RUN_LENGTH specifies the number of frames to use. Because of the nature of how the EM algorithm works, all the frames must be loaded into memory at the same time during the model-learning phase. The memory requirements for this algorithm are therefore quite high.

The parameter MAX_EM_ITERATIONS controls the maximum number of iterations performed per pixel by the EM algorithm – this is set to 50. When we ran some tests with several video sequences, it was found out that very few image pixels take more than 50 iterations to converge to a stable model. And even for these pixels, the model after 50 iterations is still adequate for motion detection.

When this algorithm was first implemented, the EM algorithm was taking about 5 to 6 minutes to learn the model from the first 700 frames (≈50 seconds). The memory requirements for loading 700 frames into memory are also very high: about 930Mb.

The algorithm was modified to use a shorter run length for doing the initial EM algorithm (100 frames; ≈8 seconds), followed by a very fast model update for the rest of the duration of the BACKGROUND_RUN_LENGTH. Running the EM algorithm on 100 frames takes much less time – around 45 seconds. This second method still generates an accurate background model. The parameter EM_RUN_LENGTH determines the actual number of frames from the BACKGROUND_RUN_LENGTH duration that is used by the EM algorithm.

The main disadvantage of this algorithm is that the number of Gaussians is fixed and must be selected beforehand. Some false positives were also generated due to the use of the RGB colour space. The processing speed is also much slower than that of the previous motion detectors. The GMM motion detector reduced the problem of holes and fragmentation in foreground objects suffered by the other detectors. But the improvement is not that large compared to the increase in processing cost.

6.13.2. NORMALISED RGB COLOUR SPACE & CHROMATIC TESTS

The motion detector of the previous section was modified to use the Normalised RGB colour space. The background model B is defined in terms of the colour components (r, g, s) as follows:

$$B = \Big(\omega_1, \ldots, \omega_K, \\ \mu_1 = < \mu_{(1,r)}, \mu_{(1,g)}, \mu_{(1,s)} >, \ldots, \mu_K = < \mu_{(K,r)}, \mu_{(K,g)}, \mu_{(K,s)} >, \\ \Sigma_1 = \mathbf{I}<\sigma_{(1,r)}^2, \sigma_{(1,g)}^2, \sigma_{(1,s)}^2>, \ldots, \Sigma_K = \mathbf{I}<\sigma_{(K,r)}^2, \sigma_{(K,g)}^2, \sigma_{(K,s)}^2> \Big)$$

The motion detection algorithm is similar, with an additional check done on the (r, g) planes of the Gaussian distributions to detect shadow and highlight. This is able to suppress sudden light variations that are not explained by the background model of the pixel, but which are still close to one of the distributions.

If a pixel x does not match any of the J background distributions, it is then projected onto the distributions' (r, g) plane by considering the colour value $<x_r, x_g, \mu_{(k,s)}>$ for each distribution k. If this colour value is within 3.5σ of one of the distributions, then we project the pixel into the brightness (s-) plane of that distribution, by considering the colour value $<\mu_{(k,r)}, \mu_{(k,g)}, x_s>$. The pixel is then marked as "shaded background" or "highlighted background" according to the following conditions:

If $x_S > \tau_{\alpha 1}\mu_{(k,s)}$	$(\tau_{\alpha 1} <= 1)$	pixel is "shaded background"
If $x_S < \tau_{\alpha 2}\mu_{(k,s)}$	$(\tau_{\alpha 2} >= 1)$	pixel is "highlighted background"

where $\tau_{\alpha 1}$ and $\tau_{\alpha 2}$ control the maximum amount of darkening of brightening in the scene and are given by parameters MAX_SHADOW_DARKENING and MAX_HIGHLIGHT_BRIGHTENING.

The main disadvantage of this motion detector is the fixed number of Gaussians. The results of tests done on movie 1 appear to be slightly better than the results of the Colour Mean and Variance detector. The number of pixels utilising more than one Gaussian to describe their background model is shown in Figure 3. The advantage of having more than one Gaussian for a certain fraction of image pixels must be weighed against the reduction in speed.

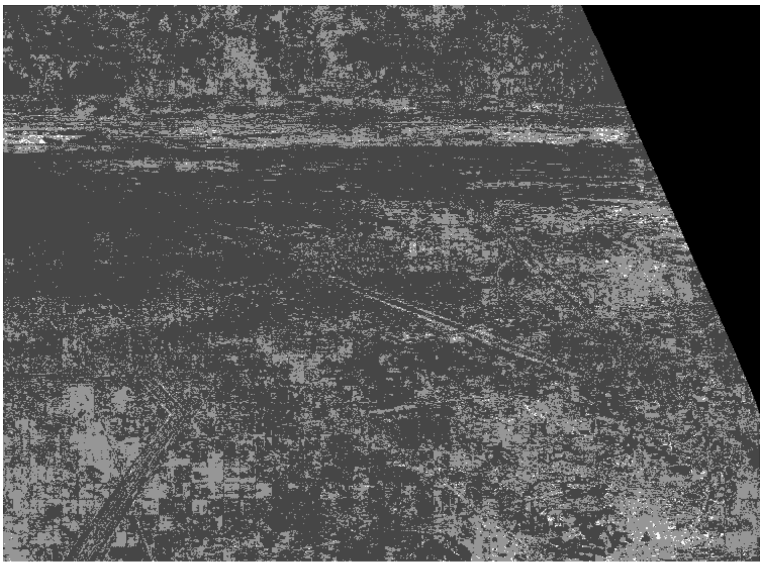


Figure 3: Pixels using 1 Gaussian are shown dark grey; Pixels with 2 Gaussians are light grey; 3 or 4 Gaussians are white. (Based on a GMM background model learnt from first 700 frames). Most of the apron pixels appear to be using multiple Gaussians because of camera noise and not real movements in the background scene.

6.14. KERNEL DENSITY ESTIMATION (KDE) METHOD

This motion detector uses the Kernel Density Estimation method described by Elgammal et al [12]. Unlike the Gaussian Mixture Model, the KDE method is more generic as it does not require the type of distribution to be specified before hand or the number of distributions. But then, since it does not use a parameterised representation, it requires large amounts of memory. It is also quite computationally expensive.

The Normalised RGB colour space (r, g, s) is used for this motion detector. The background model B consists of a window of the recent colour values for each pixel x and a covariance matrix ∑, where only the diagonal variances are non-zero (assuming independence between the colour components). The length of this window is specified by parameter BACKGROUND_RUN_LENGTH. To each value in this window, a kernel estimator function is applied, giving the required density estimation function:

$$B = \langle B_{data} = (x_1, x_2, x_3, \ldots, x_{N-1}, x_N), \Sigma = \mathbf{I}\langle \sigma_r^2, \sigma_g^2, \sigma_s^2 \rangle \rangle \qquad N = \text{BACKGROUND\_RUN\_LENGTH}$$

$$P(x) = \tfrac{1}{N}\sum_{t=1}^{N}\left(\prod_{c=r,g,s}\frac{1}{\sqrt{2\pi\sigma_c^2}}\exp\left(-\tfrac{1}{2}\frac{(x_c - x_t)^2}{\sigma_c^2}\right)\right)$$

Because the background model consists of the last N values for each pixel, a massive amount of memory is required. In this particular case, the BACKGROUND_RUN_LENGTH parameter was set to 700 frames (≈56 seconds), requiring 926Mb of memory.

A pixel x is marked as “foreground” if P(x) is smaller than the threshold PROBABILITY_THRESHOLD, which is a global parameter. Otherwise it is marked as “background”.

For pixels marked as “foreground”, an extra check is done to determine whether the pixel is a real motion pixel or not, using the chromaticity components (r, g). Given pixel x, a second probability value is calculated using those values in the background window B_{data} that have a similar brightness component to the pixel’s brightness x_s. The similarity between brightness values is determined by the parameters MAX_SHADOW_DARKENING and MAX_HIGHLIGHT_BRIGHTENING:

$$x_s > x_t \text{ MAX\_SHADOW\_DARKENING} \text{ and } x_s < x_t \text{ MAX\_HIGHLIGHT\_BRIGHTENING} \qquad (x_t \text{ in } B_{data})$$

If the probability is above the global threshold, then the pixel has the same chromaticity but different brightness and so is demoted from “foreground” and labelled “shadow or highlight”.

Updating the background model consists of simply replacing the oldest element in the background window B_{data} with the current pixel value. This in effect, makes the background window a sliding window of the last BACKGROUND_RUN_LENGTH values of a pixel.

In the current implementation of this motion detector, the background update is only performed if the pixel is marked as “background”. This selective update can cause problems – if a pixel is erroneously marked as “foreground”, it may stay in that state for a very long time. Elgammal et al [12] proposes the use of two background models – a short term and a long term model. But this has not been implemented as it would double the memory requirements.

The main advantage of the KDE motion detector is that the number and type of distributions are no longer fixed beforehand. The disadvantages are that massive amounts of memory are required and slower processing speeds. There were no improvements in the detection rate of this motion detector over the GMM one.

6.15. LINEAR PREDICTION (WALLFLOWER) METHOD

Toyama et al [13] describe a motion detection algorithm, called Wallflower, that works on 3 levels – pixel-level, region-level and frame-level. The pixel-level part of the algorithm makes use of a Linear Prediction filter that uses the recent history of a pixel x to estimate what the next value should be, as well as the expected squared prediction error $E[\varepsilon_t^2]$.

$$x_k = \sum_{t=1}^{N} \alpha_t x_{k-t} \qquad \text{using recent history} = (x_1,\ x_2,\ x_3,\ \ldots,\ x_{N-1},\ x_N).$$

$$E[\varepsilon_t^2] = E[x_k^2] + \sum_{t=1}^{N} \alpha_t\, E[x_k x_{k-t}]$$

Therefore, the background model B consist of the N prediction coefficients, a system matrix **P**, and the window of last N pixel values:

$$B_x = (\ <\alpha_1, \alpha_2, \ldots, \alpha_N>,\ \mathbf{P},\ <x_1,\ x_2,\ x_3,\ \ldots,\ x_{N-1},\ x_N>\ )$$

In [13], the size of the window of recent values is set to 50, which on image sizes of 160x120 does not require large amounts of memory. But on the 768x576 images of our test movie sequence, both memory and processing requirements are quite high. Using a shorter window size was found to adversely affect the quality of predictions. Therefore, this problem was solved in this implementation by using two models – a short-term model that has a window size N given by BACKGROUND_RUN_LENGTH (set to 10); and a long-term model that uses sparse sampling from a much larger window. The size of this long-term model is specified by the parameter LONG_TERM_MODEL_RUN_LENGTH (configured to 10 samples from a window of size 100).

$$B = (\ B_{\text{short term model}},\ B_{\text{long term model}}\ )$$

Motion is detected at pixel x, by running the linear predictors of the short-term and long-term models to get the predicted values x_p and the expected squared prediction errors. If the actual error is more than the predicted error for both of the models, then the pixel is marked as “foreground”.

$$e_{actual,Bs} = |\, x_{p,Bs} - x\, |$$
$$e_{actual,Bl} = |\, x_{p,Bl} - x\, | \qquad \text{where } B_l\text{=long-term model, } B_s\text{=short-term model}$$

If $(e_{actual,Bs} > e_{predicted,Bs})$ and $(e_{actual,Bl} > e_{predicted,Bl})$ pixel is “foreground”
Else pixel is “background”

The region-level motion detection of the Wallflower algorithm addresses the problem of uniformly coloured moving objects that exhibit no perceivable motion. This is based on finding areas of true motion using a sequence of frame differencing operations and taking the union with the last motion image. A histogram is constructed for the resulting regions and the histogram is back-projected into the current image, followed by region growing.

When region-level processing was implemented in our motion detector and tested on the video sequence, the improvement in motion detection result was marginal and not worth the extra processing cost. So, region-level processing was configured to be disabled by default.

The frame-level processing of Wallflower consists of keeping a set of background models. If more than 70% of the current image is classified as foreground, the current background model is abandoned and the background model that gives the least number of foreground pixels is used instead. The use of multiple background models was not implemented in our motion detector.

When tested on movie 1, this motion detector was found to give very good detection results. The main advantage is in detecting foreground objects that are very similar in colour to the background, so reducing the problem of fragmentation and holes in foreground objects. It was also found to be robust to illumination changes even though the RGB colour space was used. The main disadvantage is the processing cost of this algorithm. Other problems include: in the presence of sudden changes, the linear predictor can overcompensate in its prediction causing some noise (false positives) to appear in areas through which foreground objects have passed through. Also, some false positives can remain in the scene for a very long time, due to the small window size being used.

6.16. MEAN RGB

This section presents the work achieved at INRIA during the ADVISOR project. The objective of this task is to detect image changes due to people (moving objects) and to generate a description for each moving region which corresponds to the image changes. In addition, we decided to also classify the moving regions into mobile objects classes like *Person, Metro-train* because this information is very important for higher processing levels (Tracker and Behaviour Recognition module).

To reach this goal, we designed and implemented a module that contains four functionalities:

- Detection of moving regions.
- Update of the reference image.
- Classification of the detected moving regions.
- Merge of detected moving regions and reclassification.

The moving regions detected by this module are mainly the input for the tracker developed under T2.5.

6.16.1. DETECTION OF MOVING REGIONS FROM SUBTRACTION OF A REFERENCE IMAGE

The goal is to detect moving regions (defined as image changes) in the current image compared to the reference image. The reference image corresponds to the background image.

Input

- The current image (type image_t)
- The reference image (type image_t)
- An internal parameter: the threshold of the difference of intensity (type int)

Output

- The list of moving regions (type blob_t) corresponding to the detected regions.

Detection of mobile pixels

We compare all the pixels of the current image with the reference image. A pixel (x,y) is « mobile » if the difference of its intensity between the current image and the reference image is above a threshold.

pixel(x, y) = « mobile » if

$$I_{\text{difference}}(x, y) = \text{Abs}(\text{Difference}(I_{\text{current\_image}}(x, y), I_{\text{reference\_image}}(x, y))) > \textit{threshold}$$

Where $I_{\text{difference}}$ (x, y) is the intensity difference of pixel(x, y) ; $I_{\textbf{current\_image}}$ is the intensity of pixel(x, y) in the current image, $I_{\textbf{reference\_image}}$ is the intensity of pixel(x, y) in the background image**.**

This first stage consists in labelling all pixels of the current image into « mobile » or « background » (see Figure 4). To achieve a faster processing, we test the pixels at regular step: if the two pixels at consecutive steps have the same label (e.g. « background »), then we consider that all intermediary pixels (between the two previously tested) have also the same label (e.g. « background »). If it is not the case (one is

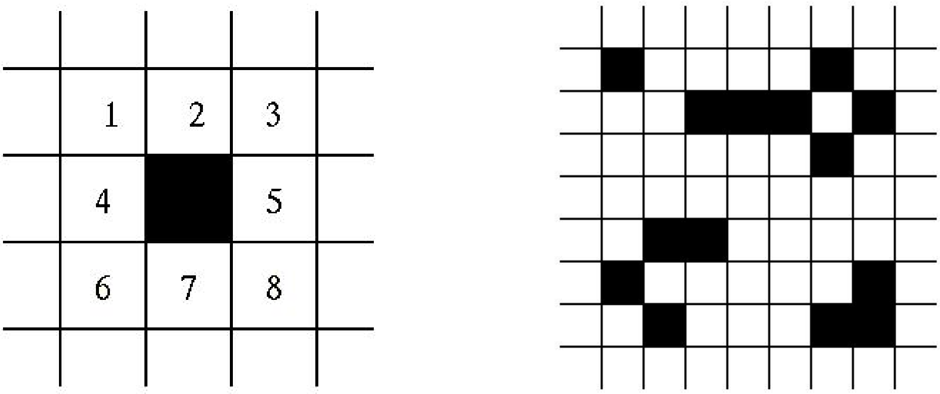
« background » and the other is « mobile »), we test directly all the intermediatery pixels. As there are less « mobile » pixels in the image, we save processing time by not testing all « background » pixels.

(a) (b)

Figure 4: *(a) raw image; (b) the black pixels are labelled "mobiles".*

Connected Regions of mobiles pixels

The second stage consists of grouping all connected pixels labelled « mobile » into connected regions. Two pixels are connected if they are eight-neighbours as shown in Figure 5.



(a) (b)

Figure 5 *(a) all these eight pixels are the eight-neighbours of the pixel in the centre. (b) On this image there are four connected regions.*

To obtain a faster processing we encode the current image using RLE (Run Length Encoded) segments. At the end of this stage we obtain an array of regions with three main characteristics:

- The bounding box (minimum and maximum of 2D position)
- The number of « mobile » pixels
- The centre of gravity of the « mobile » pixels

The last stage consists in filtering the connected regions (using their sizes, their density and their position) which are called the moving regions (see Figure 6). So only the most significant moving and connected regions are kept.

(a) (b)

Figure 6: *(a) raw image; (b) the moving regions represented by the red bounding boxes .*

6.16.2. UPDATE OF THE REFERENCE IMAGE

The reference image corresponds to the background image without mobile objects. This reference image is updated during the process to take into account the illumination changes. To update the reference image, a part of the current image which is classified as « background » is integrated in the reference image.

The reference image does not need to be updated totally at each frame. In practise, to decrease the processing time, just a part (1/5) of the reference image is updated.

<u>Input</u>

- The current image (type image_t)
- The reference image (type image_t)
- List of labelled moving regions (type blob_t)
- A set of internal parameters: coefficients of integration α and β.

<u>Output</u>

- The updated reference image (type image_t)

<u>Selection of regions with status background</u>

By default, we consider that all the image with the label « background » belongs to the reference image. In addition, some moving regions could also belong to the reference image. So the first stage consists in selecting which are the moving regions belonging to the reference image. For doing this selection, we use the label of the moving regions: all the moving region labelled as *Noise* are considered as belonging to the background of the scene. Most of the time, these regions labelled as *Noise* correspond to bad detection (e.g. shadows, reflection).

<u>Updating pixels of the reference image</u>

The second stage consists, for each pixel classified as « background » or belonging to a moving region labelled as *Noise,* of integrating the corresponding pixel of the current image into the reference image.

$$I_{reference\_image}(x, y, t) = (1-\alpha)\, I_{reference\_image}(x, y, t-1) + \alpha\, I_{current\_image}(x, y, t)$$

Where $I_{\mathbf{reference\_image}}$ (x, y, t) is the intensity of pixel(x, y) in the background image at time t ; $I_{\mathbf{reference\_image}}$ (x, y, t-1) is the intensity of pixel(x, y) in the background image at time t-1 ; $I_{\mathbf{current}\_image}$ (x, y, t) is the intensity of pixel(x, y) in the current image at time t and α is the coefficient of integration (in practice, α = 0.01).

Then, for each pixel classified as « mobile », the corresponding pixel in the current image is slightly integrated into the reference image to correct wrong classification. For example, even if a trashcan is moved from the platform and is classified as a *Person* by the Motion Detector, the trashcan will be integrated in the reference image after a certain amount of time.

$$I_{reference\_image}(x, y, t) = (1-\beta)\, I_{reference\_image}(x, y, t-1) + \beta\, I_{current\_image}(x, y, t)$$

Where $I_{\mathbf{reference\_image}}$ (x, y, t) is the intensity of pixel(x, y) in the background image at time t ; $I_{\mathbf{reference\_image}}$ (x, y, t-1) is the intensity of pixel(x, y) in the background image at time t-1 ; $I_{\mathbf{current}\_image}$ (x, y, t) is the intensity of pixel(x, y) in the current image at time t and β is the coefficient of integration (in practice, $\beta = 0.0005$).

7. CONFIGURATION PARAMETERS

The table below lists the parameters that need to be configured for each of the motion detection algorithms described in the previous sections:

Algorithm	**Parameter**	**Description**	**Value used**
Adjacent Frame Differencing	Threshold	This threshold is applied on the difference image to determine whether a pixel is foreground or background	40
Adjacent Frame Differencing & Morphology	Threshold	See above	40
	o_size	Size of the structure element used for the morphological opening operation	3x3
	c_size	Size for the structure element used for the morphological closing operation	5x5
Mean & Threshold	Threshold	See above	40
	Alpha	Learning Rate for the Running Average	0.1
	o_size	Size for the structure element used for the morphological opening operation	3x3
	c_size	Size for the structure element used for the morphological closing operation	5x5
Mean & Standard Deviation	Alpha	Learning Rate for the running average	0.1
	τ	If a pixel is beyond $\tau\sigma^2$ the mean it is detected as foreground	3
	o_size	Size for the structure element used for the morphological opening operation	3x3
	c_size	Size for the structure element used for the morphological closing operation	5x5
W^4	update_rate	Determines how often the parameters are updated	20
Frame Differencing & Gaussian	sigma	Sigma for Gaussian smoothing	0.7
	k_size	LIFO image buffer size	50
	slevel	Significance level of the Gaussians	0.25
	minbgrange	Minimum range of the background distribution	10
	alpha	Learn rate for exponential averaging the background distribution	0.1
	a_threshold	Threshold for the sigmoid function	1
	b_threshold	Threshold to determine the maximum pixel value difference which still semantically defines stationarity	40

	time	Time limit for which a significant intensity difference is noticed	10
Median & Morphology	Buffersize	Determines the size of the buffer for median calculation	100
	update_rate	Determines how often a image is added to the median buffer	40
	o_size	Size for the structure element used for the morphological opening operation	5x5 7x7
	c_size	Size for the structure element used for the morphological closing operation	20x20
Kalman Filter	A	System matrix	(1.0,0.7,0.0,0.7)
	H	Measurement matrix	(128,0,128,0,…)
	α	Learning rate foreground	0.15
	β	Learning rate background	0.25
Chromatic Background Model (*based on Kalman Filter)*	t_{CD}	Chromatic distortion threshold	12
	t_{α_low}	Lower bound threshold for the brightness distortion	-35
	$t_{\alpha1}$, $t_{\alpha2}$	Thresholds – brightness range	8 / -8
Median Filter	DETECT_DIFF_THRESHOLD	Pixels with differences above this threshold are considered as foreground.	0.2
	GAP_SIZE	Number of dilation operations	1
	BACKGROUND_RUN_LENGTH	The run length of pixel values using during background adaptation.	32
Colour Mean & Variance – RGB	HYSTERESIS_THRESHOLDING_ITERATIONS	The number of iterations used during thresholding by hysterisis.	2
	BACKGROUND_RUN_LENGTH	The first N frames used for constructing the background model.	700
	BACKGROUND_UPDATE_WEIGHT	The learning rate used for background adaptation for pixels marked as “background”	0.005
	FOREGROUND_UPDATE_WEIGHT	The learning rate used for background adaptation for pixels marked as “foreground”.	0.0002
	MIN_SIGMA	The standard deviation is set to this minimum value if it becomes 0 after the background update.	1.0
Colour Mean & Variance – Normalised RGB & Chromatic Tests	HYSTERESIS_THRESHOLDING_ITERATIONS	See previous algorithm.	2
	BACKGROUND_RUN_LENGTH	See previous algorithm.	700
	BACKGROUND_UPDATE_WEIGHT	See previous algorithm.	0.005
	FOREGROUND_UPDATE_WEIGHT	See previous algorithm.	0.0002
	MIN_SIGMA	See previous algorithm.	1.0
	MAX_SHADOW_DARKENING	The maximum amount of brightness reduction that can occur in a shaded background.	0.65
	MAX_HIGHLIGHT_BRIGHTENING	The maximum amount of brightness increase that can occur due to a highlighted background.	1.4

Colour Mean & Variance – HSV & Chromatic Tests	HYSTERESIS_THRESHOLDING_ITERATIONS	See previous algorithm.	2
	BACKGROUND_RUN_LENGTH	See previous algorithm.	700
	BACKGROUND_UPDATE_WEIGHT	See previous algorithm.	0.005
	FOREGROUND_UPDATE_WEIGHT	See previous algorithm.	0.0002
	MIN_SIGMA	See previous algorithm.	1.0
	MAX_SHADOW_DARKENING	See previous algorithm.	0.6
	MAX_HIGHLIGHT_BRIGHTENING	See previous algorithm.	1.4
	LOW_SV_THRESHOLD	HSV colours with S or V below this value have an unstable hue H.	0.18
Colour & Gradient Fusion	HYSTERESIS_THRESHOLDING_ITERATIONS	See previous algorithm.	2
	BACKGROUND_RUN_LENGTH	See previous algorithm.	700
	BACKGROUND_UPDATE_WEIGHT	See previous algorithm.	0.005
	FOREGROUND_UPDATE_WEIGHT	See previous algorithm.	0.0002
	MIN_SIGMA	See previous algorithm.	1.0
	MAX_SHADOW_DARKENING	See previous algorithm.	0.65
	MAX_HIGHLIGHT_BRIGHTENING	See previous algorithm.	1.4
	MIN_EDGE_SIGMA	The edge model standard deviation is set to this minimum value if it is less than this in the background model.	4.0
Gaussian Mixture Model – RGB	NUM_GMM_DISTRIBUTIONS	Number of Gaussians in the mixture model.	4
	MAX_BACKGROUND_WEIGHT	The first j distributions having a combined weight smaller than this parameter are considered to represent the background scene.	0.85
	NUM_SIGMA_THRESHOLD	Values that are within this number of standard deviations from the distribution are considered to belong to the distribution	3.5
	BACKGROUND_RUN_LENGTH	See previous algorithm.	700
	EM_RUN_LENGTH	The number of initial frames from the BACKGROUND_RUN_LENGTH that are used by the EM algorithm during model learning.	100
	MAX_EM_ITERATIONS	The maximum number of iterations.	50
	BACKGROUND_UPDATE_WEIGHT	See previous algorithm.	0.003
	FOREGROUND_UPDATE_WEIGHT	See previous algorithm.	0.0002
Gaussian Mixture Model – Normalised RGB & Chromatic Tests	NUM_GMM_DISTRIBUTIONS	See previous algorithm.	4
	MAX_BACKGROUND_WEIGHT	See previous algorithm.	0.85
	NUM_SIGMA_THRESHOLD	See previous algorithm.	3.5
	BACKGROUND_RUN_LENGTH	See previous algorithm.	700
	EM_RUN_LENGTH	See previous algorithm.	100
	MAX_EM_ITERATIONS	See previous algorithm.	50
	BACKGROUND_UPDATE_WEIGHT	See previous algorithm.	0.0015
	FOREGROUND_UPDATE_WEIGHT	See previous algorithm.	0.0001
	MAX_SHADOW_DARKENING	See previous algorithm.	0.7

	MAX_HIGHLIGHT_BRIGHTENING	See previous algorithm.	1.2
Kernel Density Estimation	BACKGROUND_RUN_LENGTH	This is the length of the data window used for the background model. This is a sliding window.	700
	PROBABILITY_THRESHOLD	Probability threshold for the density estimation.	1e-12
	MAX_SHADOW_DARKENING	See previous algorithm.	0.7
	MAX_HIGHLIGHT_BRIGHTENING	See previous algorithm.	1.2
Linear Prediction (Wallflower)	BACKGROUND_RUN_LENGTH	The run-length for the short-term model data window.	10
	LONG_TERM_MODEL_RUN_LENGTH	The run-length for the long-term model data window.	10
	MAX_PREDICTED_ERROR_FACTOR	Maximum expected prediction error compared to the pixel's actual error.	12.0
	MODEL_UPDATE_WEIGHT	adaptation rate of the model to new data	0.5
Mean RGB	Threshold	The threshold of the difference of intensity	
	α	Coefficient of integration for pixels classified as background	0.01
	β	Coefficient of integration for pixels classified as mobile	0.0005

8. EVALUATION OF THE MOTION DETECTORS

8.1. ADVANTAGES AND DISADVANTAGES

The table below lists the advantages and disadvantages of the motion detection algorithms:

Algorithm	Advantages	Disadvantages
Adjacent Frame Differencing	• Simple	• no long time background model • only outline detection • global constant threshold • high amount of false negatives • insufficient detection rate
Adjacent Frame Differencing & Morphology	• simple	• no long time background model • global constant threshold • high amount of false negatives • insufficient detection rate
Mean & Threshold		• global constant threshold • insufficient detection rate
Mean & Standard Deviation	• absence of constant threshold	• insufficient detection rate
W^4	• absence of constant threshold	• noised results • insufficient detection rate • discarding of RGB-information • high number of false positives

Frame Differencing & Gaussian		• not robust against noise • designed for detection of long term changes • insufficient detection rate
Median & Morphology	• adaptive threshold • acceptable detection results	• fading out
Kalman Filter	• adaptive threshold • acceptable detection results	• fading out • problems in detecting small, slow moving objects
Chromatic Background Model	• robust against illumination changes	• global thresholds • reflection problems • detection rate depends on the scene colours
Median Filter	• Simple • Very fast • Very short background learning	• Global threshold, manually selected • Blind background update • High amount of false negatives • Slow moving objects leave trails behind • Fragmented and noisy motion image • Not usable
Colour Mean & Variance – RGB	• Fast • Not using global detection threshold	• False positives due to sudden/large light changes • Holes and fragmentation in objects with same colour as background
Colour Mean & Variance – Normalised RGB & Chromatic Tests	• Fast • Not using global detection threshold • Handles sudden illumination changes	• Using global constants • Holes and fragmentation in objects with same colour as background • Detects strong shadows as foreground
Colour Mean & Variance – HSV & Chromatic Tests	• Fast • Not using global detection threshold • Handles sudden illumination changes	• Using global constants • Holes and fragmentation in objects with same colour as background • Detects strong shadows as foreground • Hue is susceptible to JPEG noise
Colour & Gradient Fusion	• Not using global detection threshold • Handles sudden illumination changes • Better detection rate • Less holes and fragmentation in objects	• Very slow • Using global constants • Detects strong shadows as foreground
Gaussian Mixture Model – RGB	• Not using global detection threshold • Less holes and fragmentation in objects • Handles pixels on background edges	• Slow • Fixed number of Gaussians • Using global constants • Very slow background learning • False positives due to sudden/large light changes • Detects strong shadows as foreground

Gaussian Mixture Model – Normalised RGB & Chromatic Tests	• Not using global detection threshold • Handles sudden illumination changes • Less holes and fragmentation in objects • Handles pixels on background edges	• Slow • Fixed number of Gaussians • Using global constants • Very slow background learning • Detects strong shadows as foreground
Kernel Density Estimation	• Not using global detection threshold • Handles sudden illumination changes • Less holes and fragmentation in objects • Handles pixels on background edges	• Very slow • Massive memory requirements • Using global constants • Foreground objects can remain for long time; background not updated • Detects strong shadows as foreground
Linear Prediction (Wallflower)	• Short background learning • Handles sudden illumination changes • Very good detection rate for low-contrast motion • Less holes and fragmentation in objects	• Very Slow • Using global constants • No improvements with region-level processing • Long lasting false positives • Moving objects can leave some noise behind
Mean RGB	• Robust to illumination changes • Fast	• Using threshold • Incorrect background integration for badly classified objects.

8.2. INITIAL ANALYSIS

From a first analysis of the motion detection algorithms performed by the respective teams, and expressed in the above table as advantages and disadvantages, some of the algorithms were eliminated. The algorithms and the reasons for their elimination are given below:

- **Adjacent Frame Differencing, Adjacent Frame Differencing & Morphology**

  Adjacent Frame Differencing, even with the improvement of morphological operations, does not yield acceptable results. The problem of homogenously coloured slow moving objects (only outlines are detected) and the absence of a longer term background model (which yields in a insufficient detection rate) as well as the presence of a global constant threshold are the main causes for rejecting these algorithms.

- **Mean & Threshold, Mean & Standard Deviation**

  Mean & Morphology results in insufficient detection rate, which even worsens if the standard deviation is used instead of a global constant.

- **W^4**

  W^4 produces an intolerable amount of false positives.

- **Frame Differencing and Gaussian**

  Frame Differencing and Gaussian is not robust against noise.

- **Median Filter**

  The Median Filter, with its global threshold and high error rate does not appear to be useful as a motion detector.

- **Colour Mean and Variance – RGB and HSV colour spaces**

  The versions of the Colour Mean and Variance algorithms that use the Normalised RGB and HSV space and the chromatic/brightness information give better results than the one using the RGB colour space. Normalised RGB appears to give a better result than HSV, because the latter is more affected by JPEG noise due to the instability of the hue component for low values of V and S. For these reasons, it was decided to use the Normalised RGB colour space for this algorithm and eliminate the ones using RGB and HSV.

- **Gaussian Mixture Model – RGB colour space**

  Like the previous, the version of the Gaussian Mixture Model using RGB colour space gives inferior results when compared to the version using Normalised RGB. Thus, the RGB version is eliminated.

In addition, it was observed that the **Colour and Edge Fusion** method, the **Linear Prediction** method, and to a lesser extent the **Kernel Density Estimation** method, do not satisfy the requirement of operating in real-time. These algorithms were not eliminated at this stage, because their motion detection results look promising and their use in the future may become feasible given hardware advances, etc.

After this first round of elimination, the remaining short list of algorithms consists of:

1.	**Median and Morphology**
2.	**Kalman Filter**
3.	**Chromatic Background Model**
4.	**Colour Mean and Variance** (Normalised RGB & chromatic tests)
5.	**Colour and Edge Fusion**
6.	**Gaussian Mixture Model** (Normalised RGB & chromatic tests)
7.	**Kernel Density Estimation**
8.	**Linear Prediction (Wallflower)**
9.	**Mean RGB**

8.3. EVALUATION CRITERIA

At the PRIP meeting in Vienna that occurred on 01-02 July 2004, it was decided to create a list of evaluation criteria to compare the motion detection algorithms against [14]. These criteria are listed below, and will be used to determine which is/are the best algorithm(s) for the AVITRACK project:

- Susceptibility to Noise
- Robustness to Illumination changes
- Sensitivity
- Speed

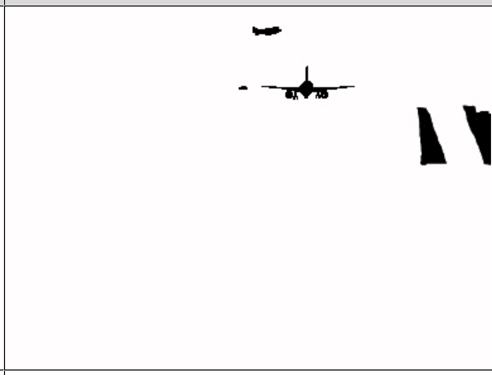
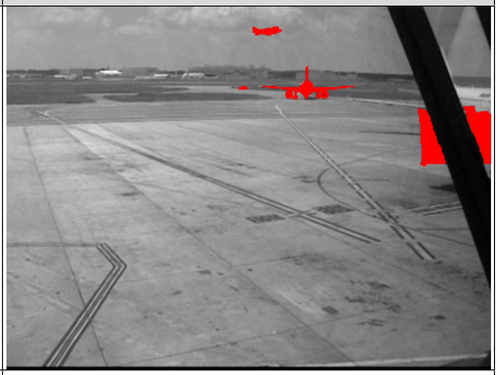
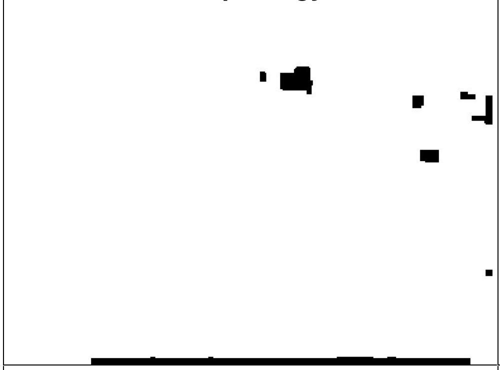
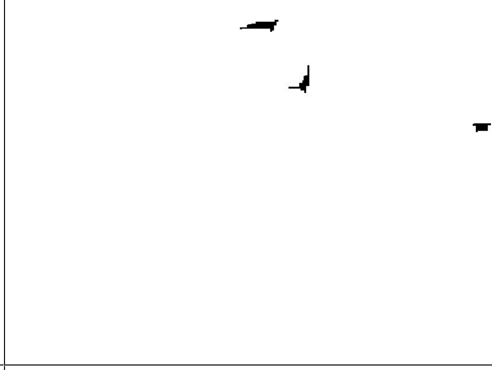
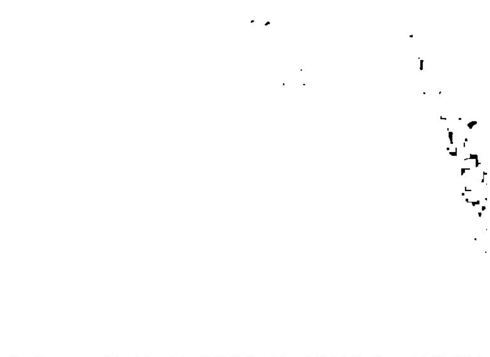
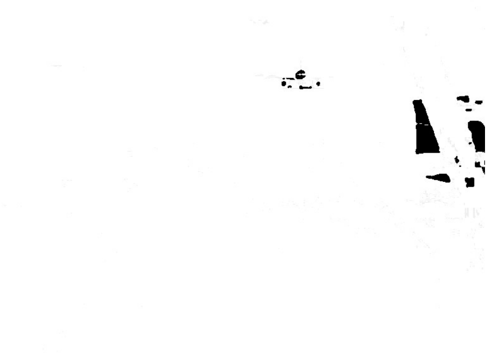
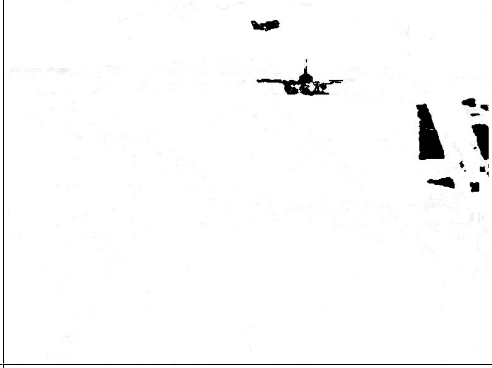
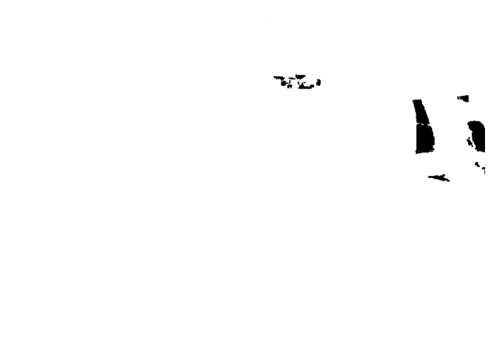
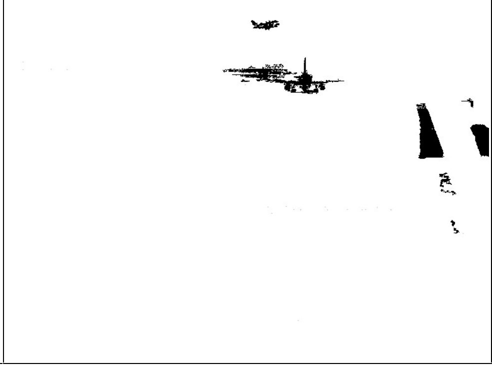
The results of each motion detector were stored as a set of motion images, with pixels segmented into foreground or background. Because of varying lengths of the background construction phase from one algorithm to the other, and since no interesting motion occurs in the first 750 frames, it was decided to start from frame 900 onwards and taking every 400th frame of the sequence for evaluation. Some other frames containing interesting activity were also added. These motion detection results are shown over the next few pages (foreground pixels displayed as black; background pixels are white).

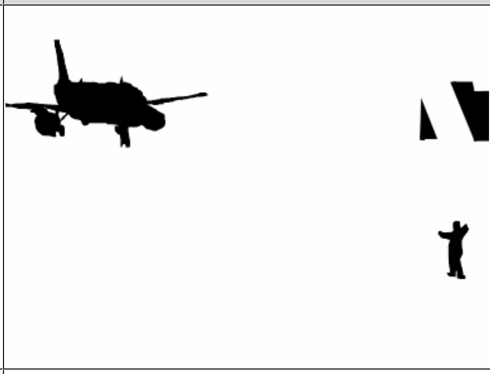
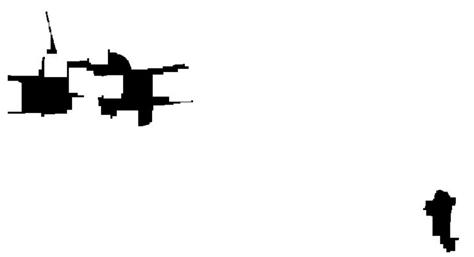
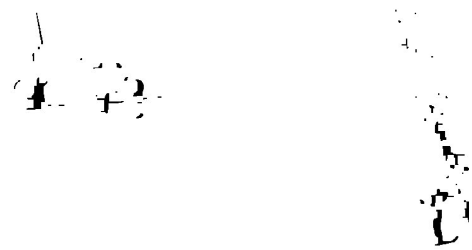
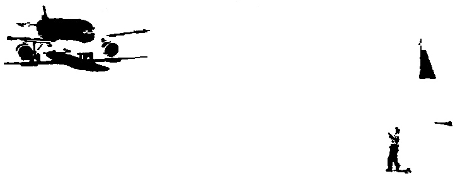
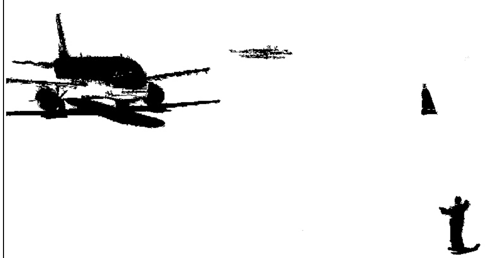
Some of the motion detection algorithms do not have a mechanism to incorporate stationary foreground objects into the background and to re-activate them when they start moving again. These algorithms are marked by the comment “no integration into background” in the figures over the next few pages, to differentiate them from others that do.

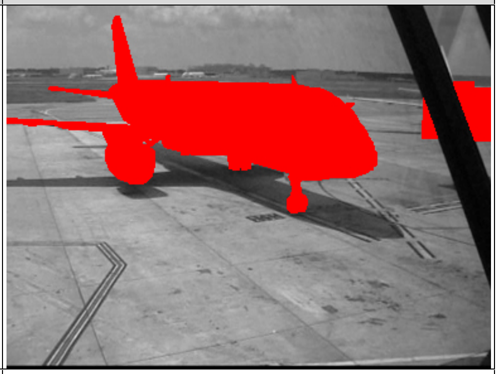
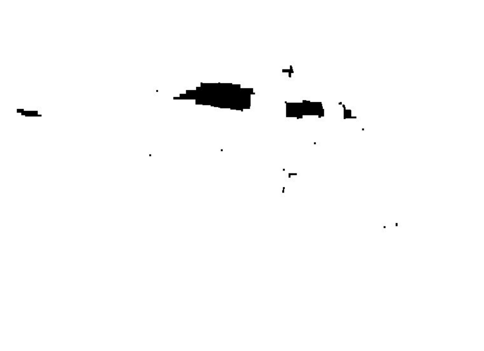
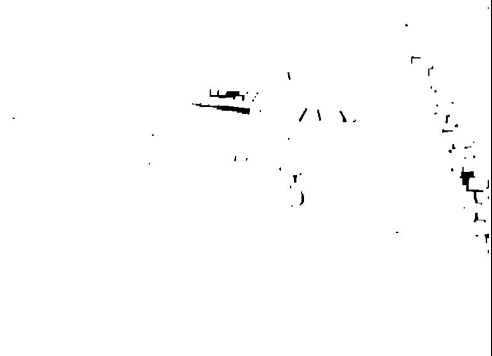
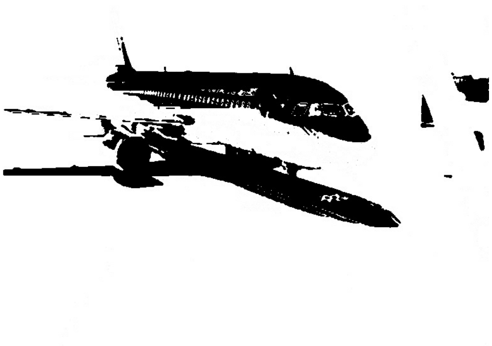
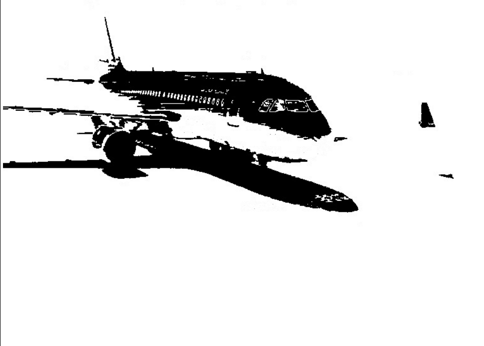
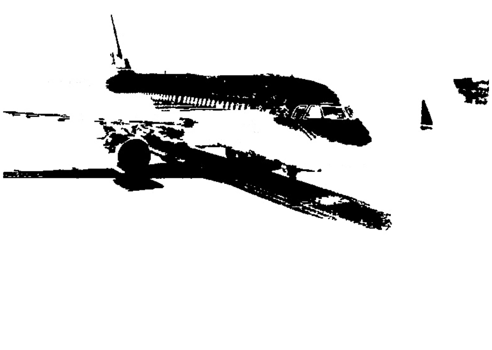
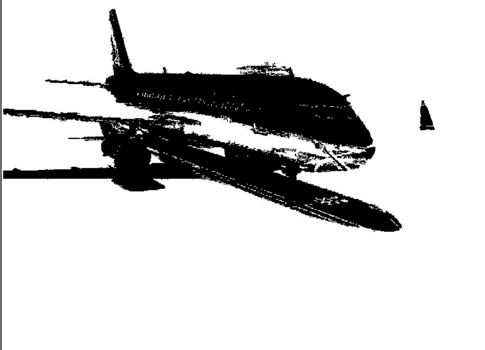
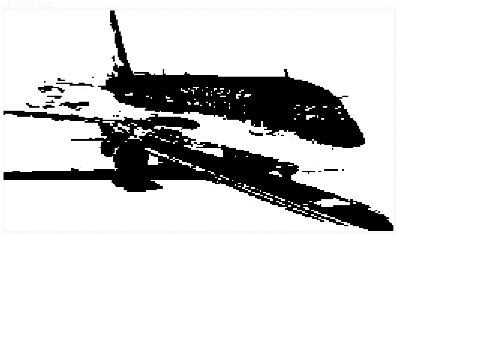
Another set of algorithms makes use of a simple form of integration into the background by checking any movement in the region’s bounding box from one frame to the next. This method is not so accurate as when using information returned by the higher-level scene tracking algorithms, which can use a much longer history to determine if a foreground object has become stationary or not. It is also prone to errors and small variations introduced by the post-processing operations (example, morphological operations) performed by these algorithms. Hence, they fail to integrate into the background all the foreground objects that have become stationary. These algorithms are marked by the comment “using simple integration method”.

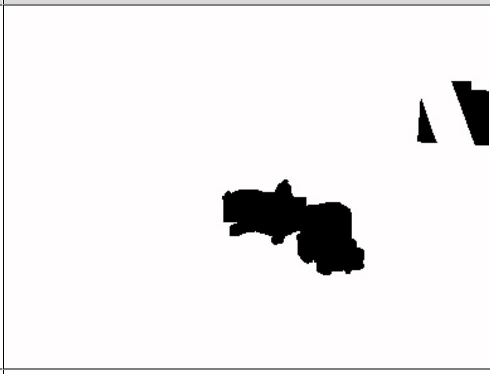
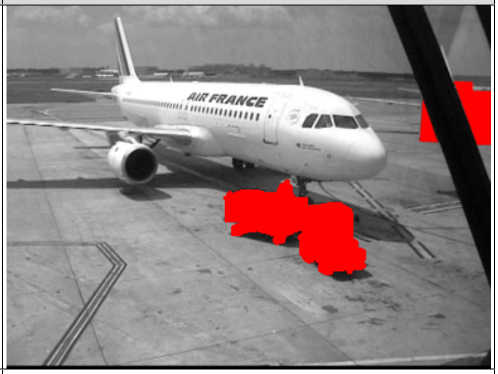
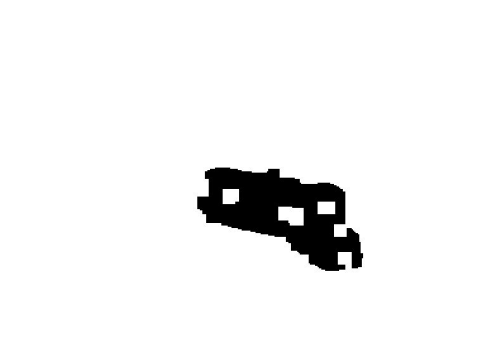
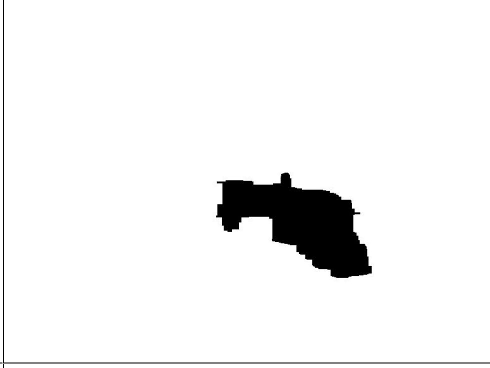
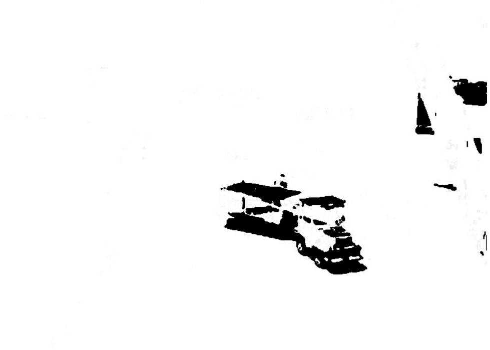
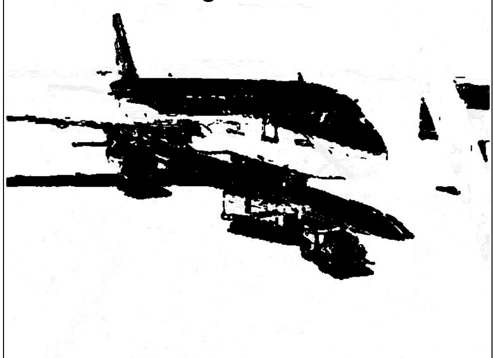
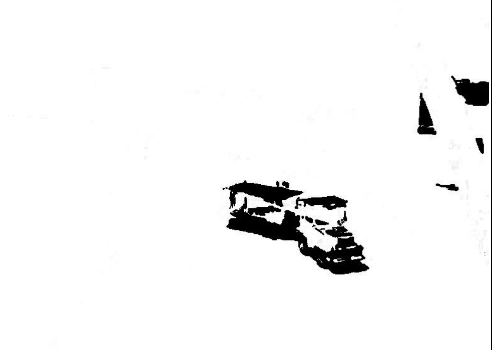
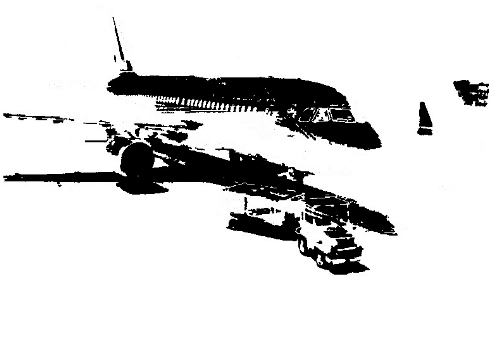
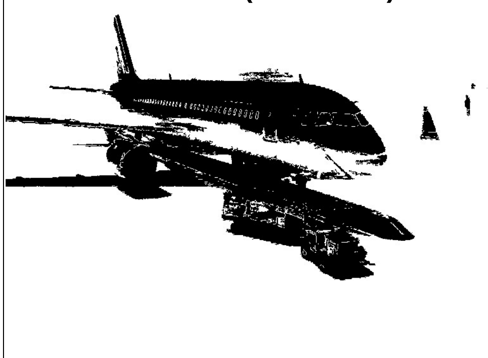
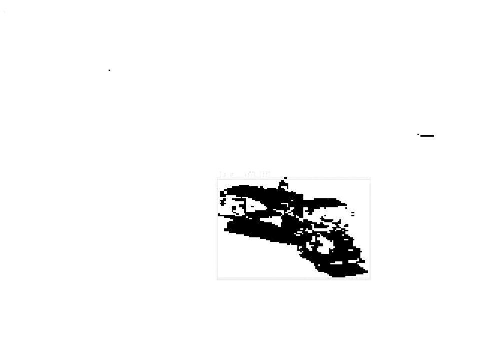
Both these cases have to be taken into consideration when comparing the results of the motion detectors in order to make a fair comparison. A mechanism to integrate objects into the background can be added later on to these algorithms, based on more accurate feedback returned by the scene tracking algorithms.

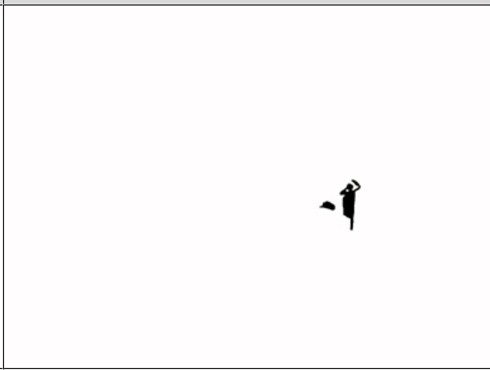
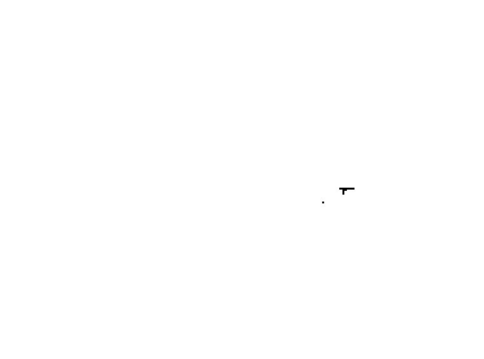
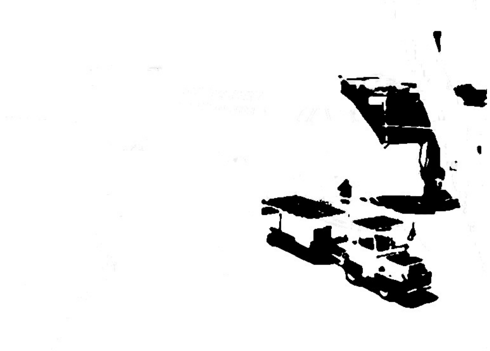
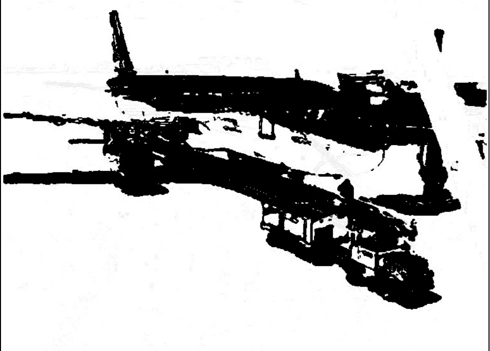
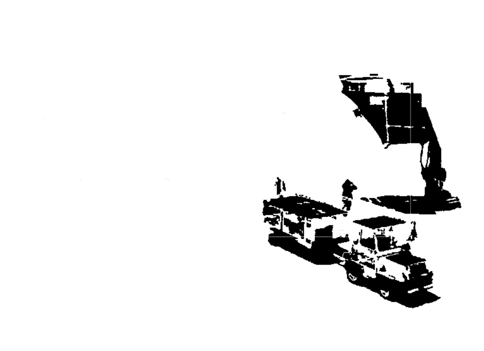
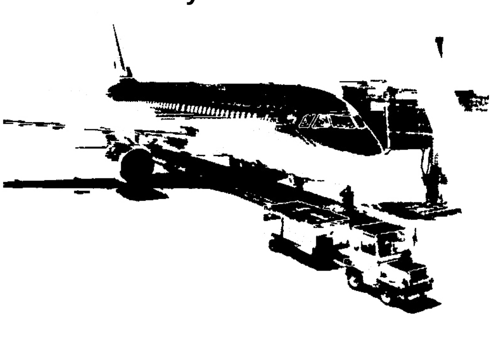
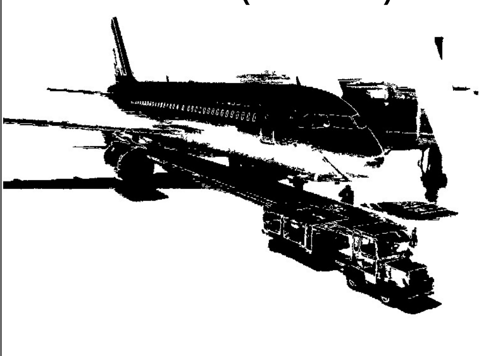
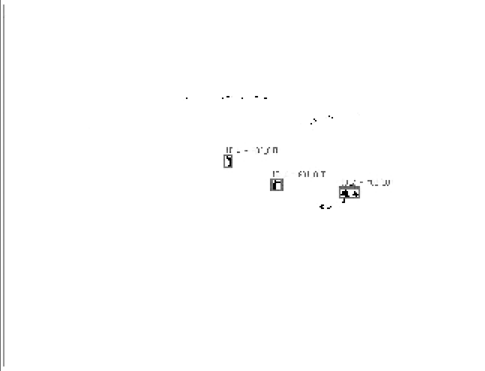
The ground truth images were created manually by examining the previous frames and looking for any motion. Objects that had moved during the last few frames where then fully segmented out as foreground objects. Shadow regions were not included in the ground truth. The ground truth images are also shown overlaid over the original image to show the motion within the context of the image.

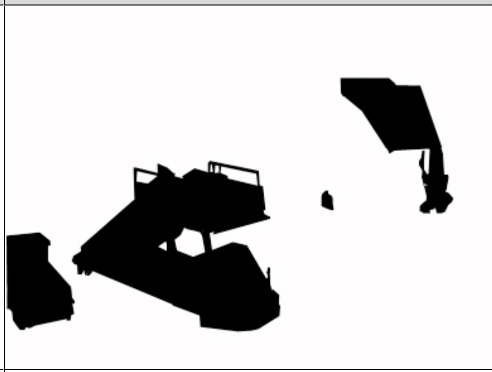
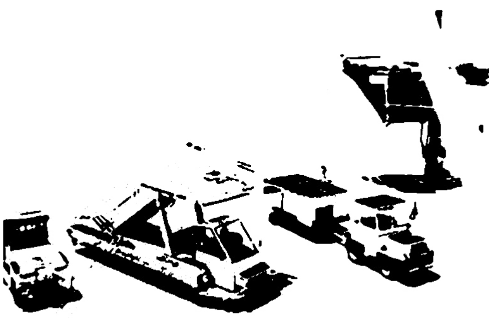
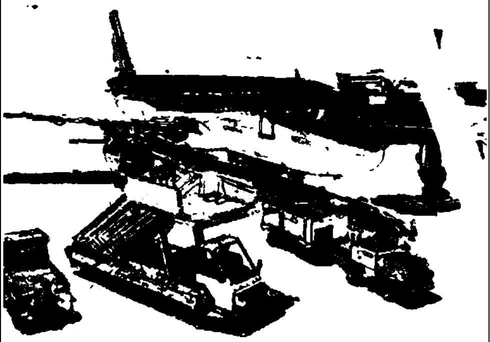
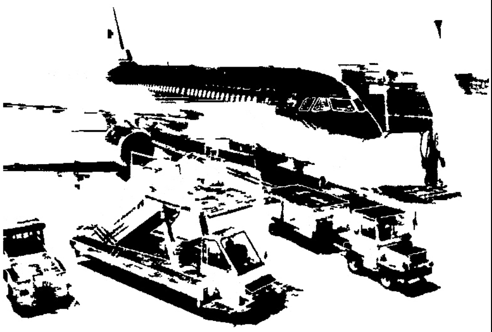
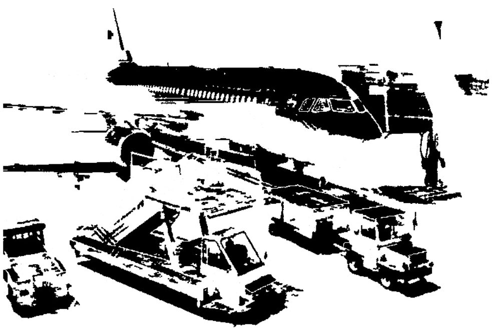
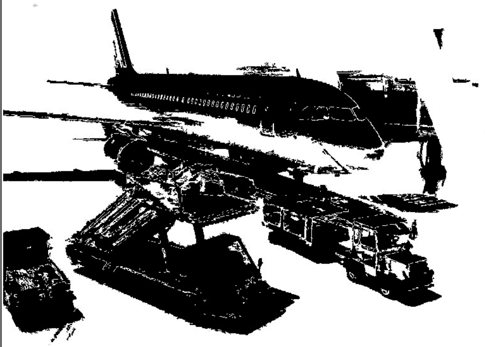
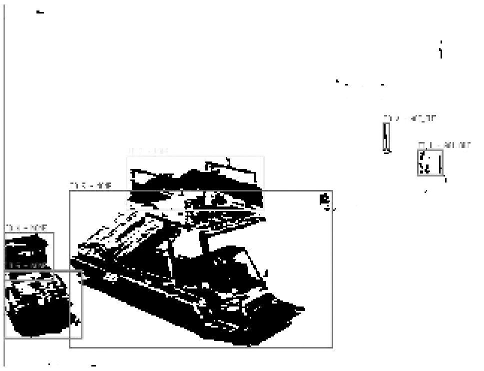
Frame # 900	**Ground Truth**	**Overlaid Ground Truth**
		
Median and Morphology 	**Kalman Filter** 	**Chromatic Background Model** 
Colour Mean and Variance 	**Colour and Edge Fusion** 	**Gaussian Mixture Model**
Kernel Density Estimation 	**Linear Prediction (Wallflower)** 	**Mean RGB**

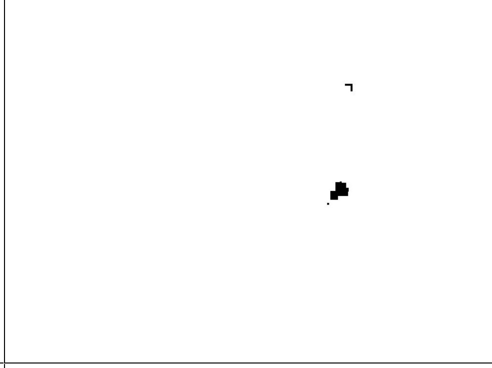
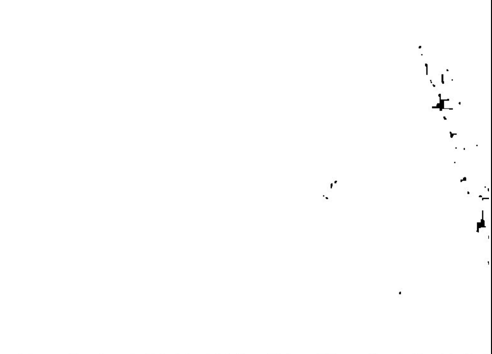
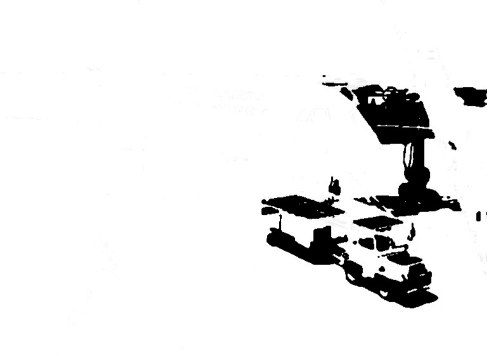
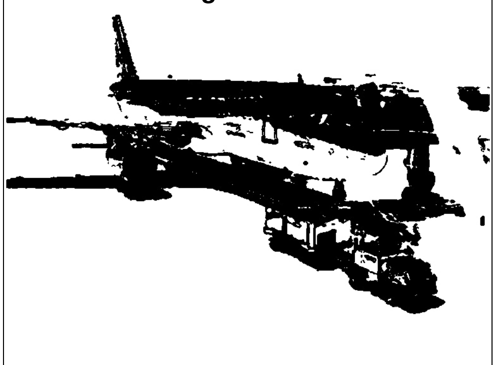
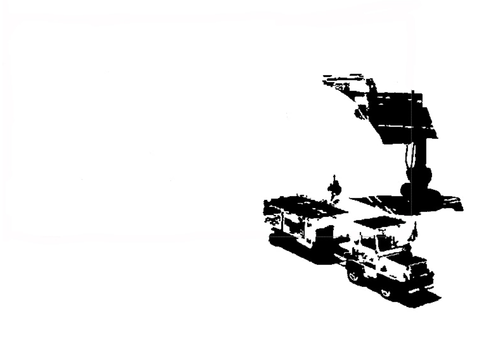
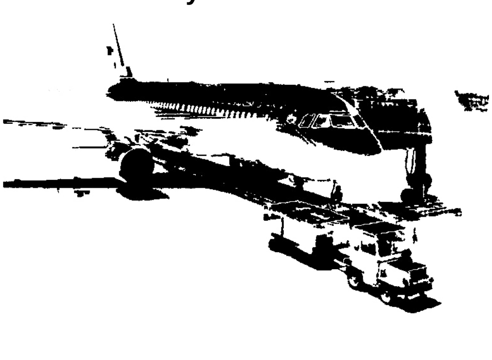
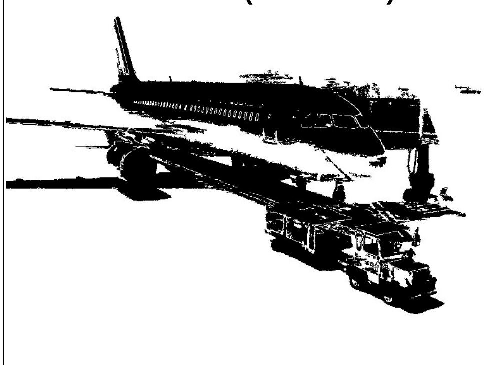
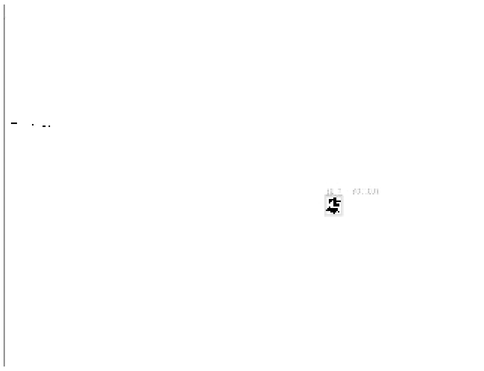
Frame # 1300	**Ground Truth**	**Overlaid Ground Truth**
		
Median and Morphology 	**Kalman Filter** 	**Chromatic Background Model** 
Colour Mean and Variance 	**Colour and Edge Fusion** 	**Gaussian Mixture Model** 
Kernel Density Estimation 	**Linear Prediction (Wallflower)** 	**Mean RGB** 

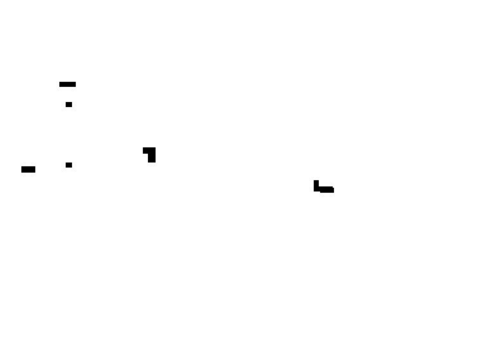
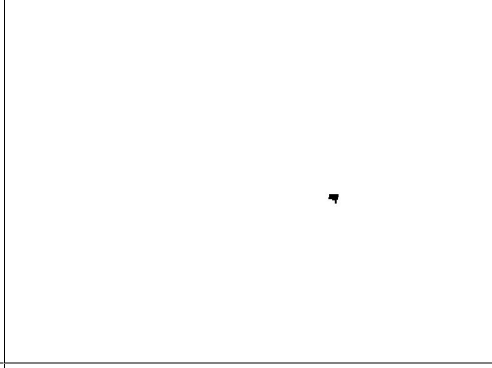
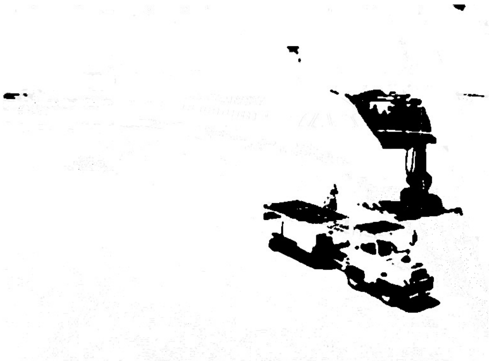
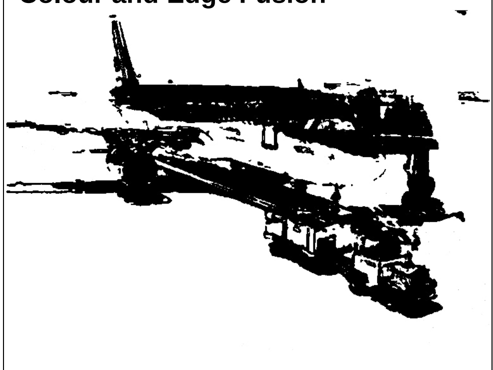
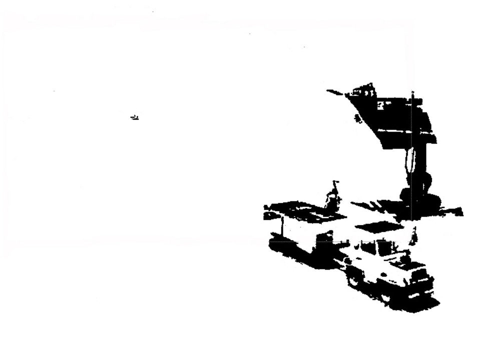
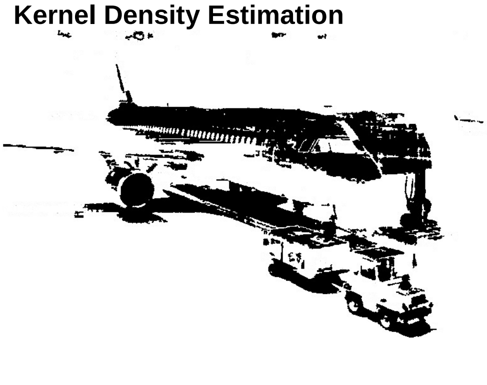
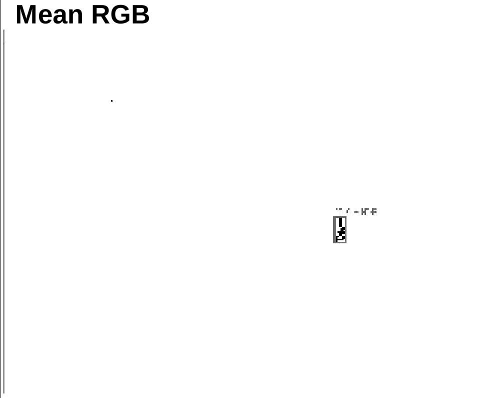
Frame # 1500	Ground Truth	Overlaid Ground Truth
		
Median and Morphology 	**Kalman Filter** 	**Chromatic Background Model** 
Colour Mean and Variance 	**Colour and Edge Fusion** 	**Gaussian Mixture Model** 
Kernel Density Estimation 	**Linear Prediction (Wallflower)** 	**Mean RGB** 

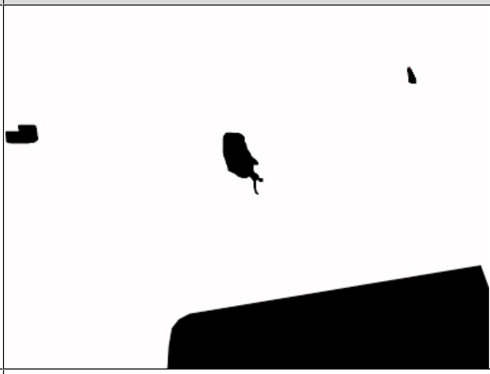
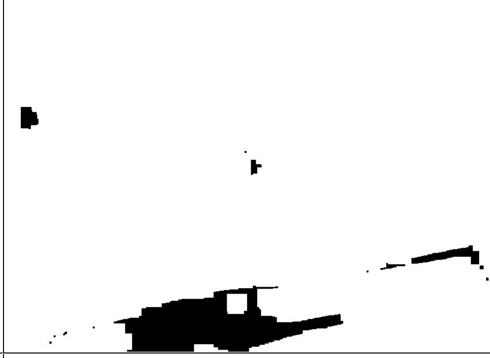
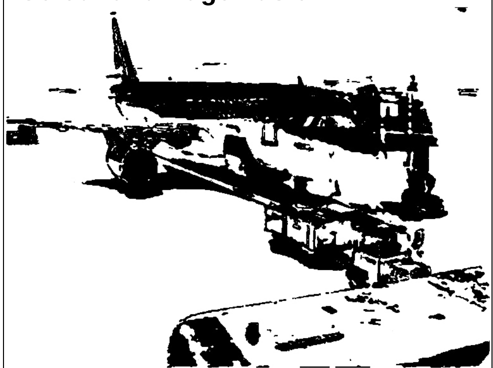
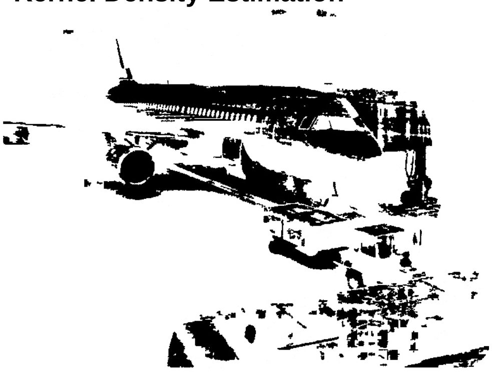
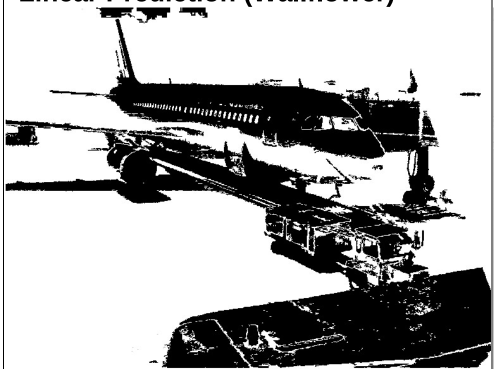
Frame # 1700	Ground Truth	Overlaid Ground Truth
		
Median and Morphology 	**Kalman Filter** 	**Chromatic Background Model** 
Colour Mean and Variance  (using simple background integration method)	**Colour and Edge Fusion**  (no integration into background)	**Gaussian Mixture Model**  (using simple background integration method)
Kernel Density Estimation  (no integration into background)	**Linear Prediction (Wallflower)**  (no integration into background)	**Mean RGB** 

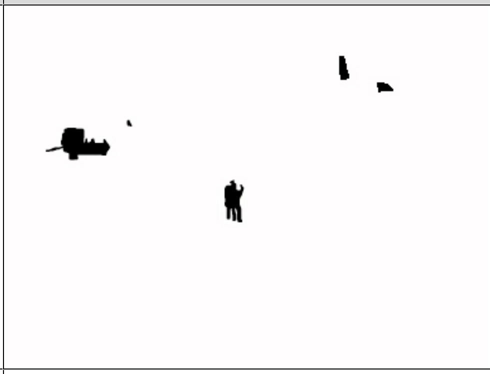
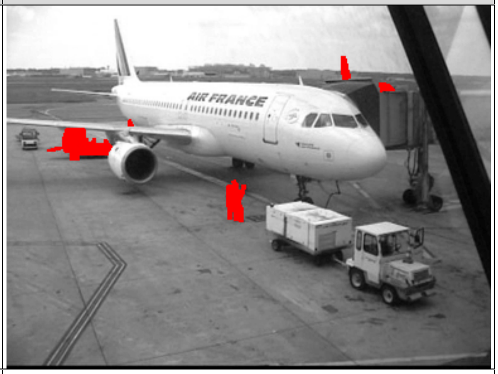
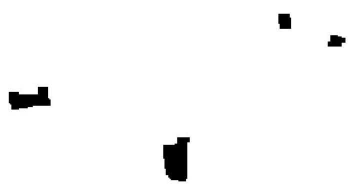
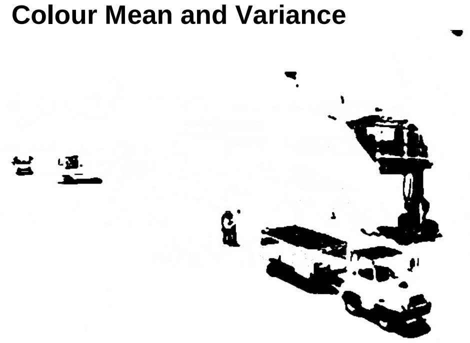
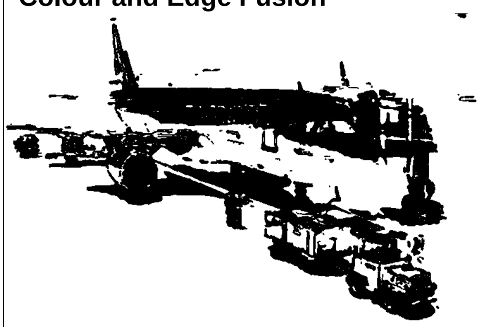
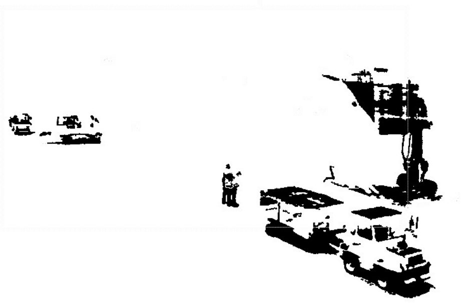
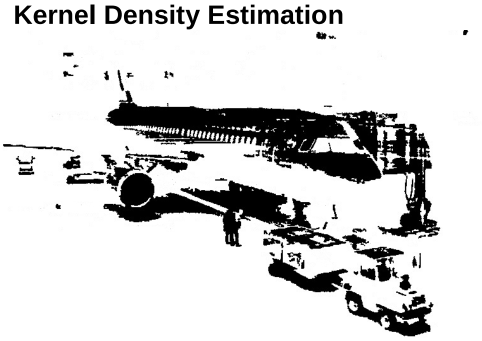
Frame # 2100	Ground Truth	Overlaid Ground Truth
		
Median and Morphology 	**Kalman Filter** 	**Chromatic Background Model** 
Colour Mean and Variance  (using simple background integration method)	**Colour and Edge Fusion**  (no integration into background)	**Gaussian Mixture Model**  (using simple background integration method)
Kernel Density Estimation  (no integration into background)	**Linear Prediction (Wallflower)**  (no integration into background)	**Mean RGB** 

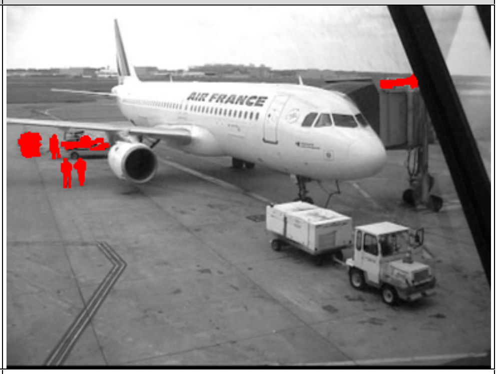
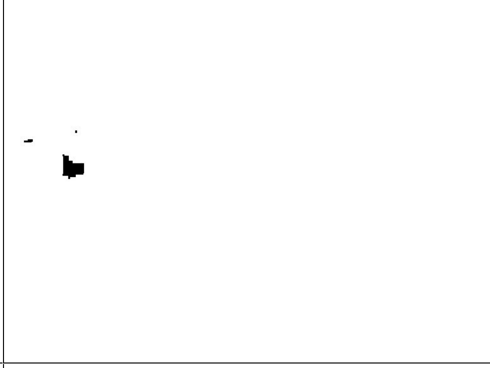
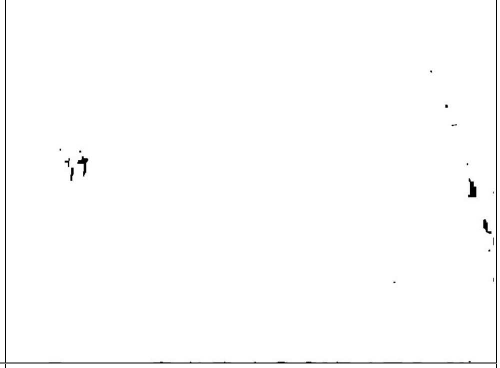
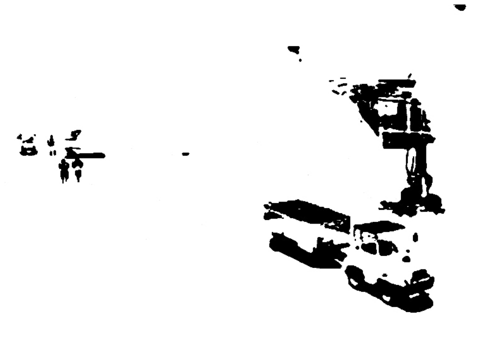
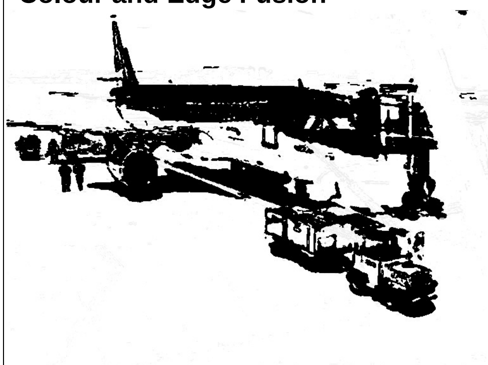
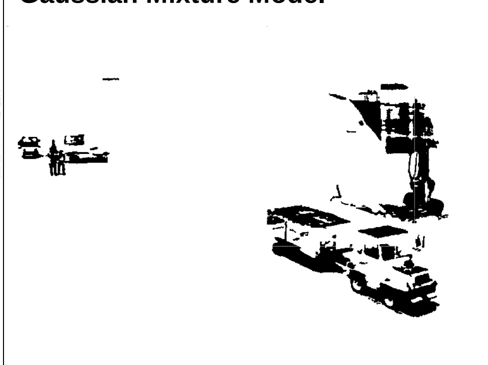
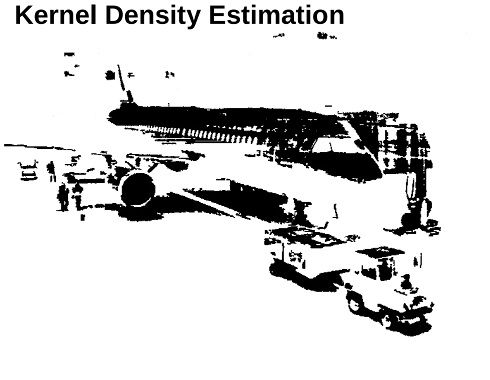
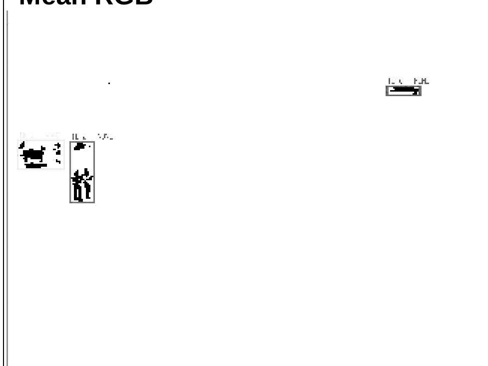
Frame # 2175	Ground Truth	Overlaid Ground Truth
		
Median and Morphology 	**Kalman Filter** 	**Chromatic Background Model** 
Colour Mean and Variance  (using simple background integration method)	**Colour and Edge Fusion**  (no integration into background)	**Gaussian Mixture Model**  (using simple background integration method)
Kernel Density Estimation  (no integration into background)	**Linear Prediction (Wallflower)**  (no integration into background)	**Mean RGB** 

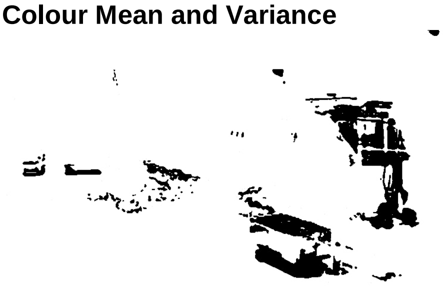
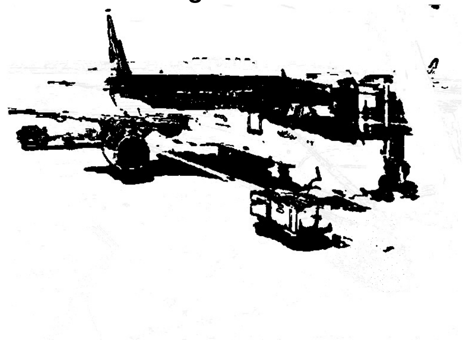
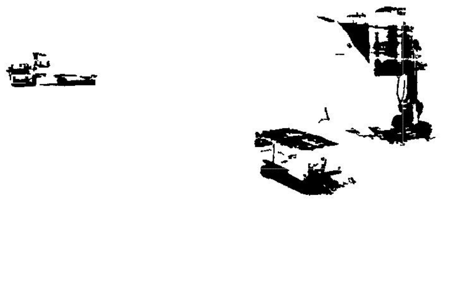
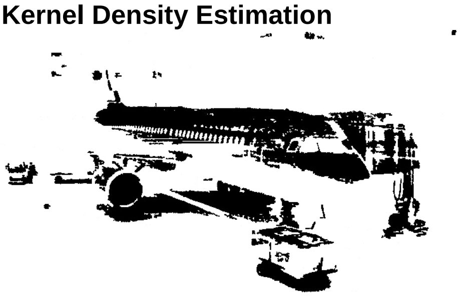
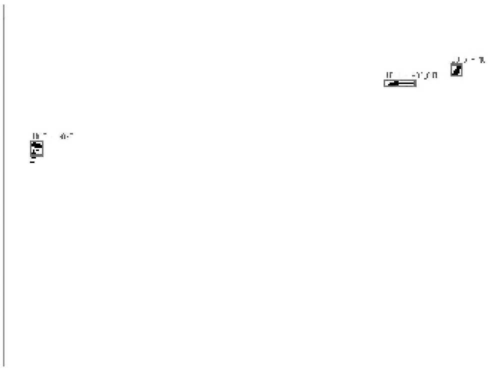
Frame # 2500	**Ground Truth**	**Overlaid Ground Truth**
		
Median and Morphology 	**Kalman Filter** 	**Chromatic Background Model** 
Colour Mean and Variance  (using simple background integration method)	**Colour and Edge Fusion**  (no integration into background)	**Gaussian Mixture Model**  (using simple background integration method)
Kernel Density Estimation  (no integration into background)	**Linear Prediction (Wallflower)**  (no integration into background)	**Mean RGB** 

Frame # 2900	Ground Truth	Overlaid Ground Truth
		
Median and Morphology 	**Kalman Filter** 	**Chromatic Background Model** 
Colour Mean and Variance  (using simple background integration method)	**Colour and Edge Fusion**  (no integration into background)	**Gaussian Mixture Model**  (using simple background integration method)
Kernel Density Estimation  (no integration into background)	**Linear Prediction (Wallflower)**  (no integration into background)	**Mean RGB** 

Frame # 3300	Ground Truth	Overlaid Ground Truth
		
Median and Morphology 	**Kalman Filter** 	**Chromatic Background Model** 
Colour Mean and Variance  (using simple background integration method)	**Colour and Edge Fusion**  (no integration into background)	**Gaussian Mixture Model**  (using simple background integration method)
Kernel Density Estimation  (no integration into background)	**Linear Prediction (Wallflower)**  (no integration into background)	**Mean RGB** 

Frame # 3700	**Ground Truth**	**Overlaid Ground Truth**
		
Median and Morphology 	**Kalman Filter** 	**Chromatic Background Model** 
Colour Mean and Variance  (using simple background integration method)	**Colour and Edge Fusion**  (no integration into background)	**Gaussian Mixture Model**  (using simple background integration method)
Kernel Density Estimation  (no integration into background)	**Linear Prediction (Wallflower)**  (no integration into background)	**Mean RGB** 

Frame # 4100	**Ground Truth**	**Overlaid Ground Truth**
		
Median and Morphology 	**Kalman Filter** 	**Chromatic Background Model** 
Colour Mean and Variance  (using simple background integration method)	**Colour and Edge Fusion**  (no integration into background)	**Gaussian Mixture Model**  (using simple background integration method)
Kernel Density Estimation  (no integration into background)	**Linear Prediction (Wallflower)**  (no integration into background)	**Mean RGB** 

Frame # 4500	Ground Truth	Overlaid Ground Truth
		
Median and Morphology	**Kalman Filter**	**Chromatic Background Model** 
Colour Mean and Variance  (using simple background integration method)	**Colour and Edge Fusion**  (no integration into background)	**Gaussian Mixture Model**  (using simple background integration method)
Kernel Density Estimation  (no integration into background)	**Linear Prediction (Wallflower)**  (no integration into background)	**Mean RGB** 

Because of the different approaches taken for some of the motion detectors, it was not possible to perform a quantitative analysis of the results against manually-segmented ground truth results, and measuring the amount of false positives and false negatives. Some of the motion detectors use a very fast background update rate while some others incorporate foreground objects into the background model when they have stopped moving. Instead of a quantitative analysis, a qualitative evaluation is performed using the above criteria. This evaluation process was initiated at the PRIP meeting and the outcome is shown in the table below:

Algorithm	Susceptibility to Noise	Robustness to Illumination Changes	Sensitivity	Speed	Comments
Median and Morphology	Acceptable; some noise present.	Robust to most illumination changes.	Acceptable; occasionally some holes.		Using Adaptive Threshold. Fading out problem.
Kalman Filter	More susceptible.	Robust to most illumination changes.		Linux: real time.	Using Adaptive thresholding. Fading out problem.
Chromatic Background Model	Acceptable.	Robust to most illumination changes.		Matlab; to be re-implemented in C++	Problem with low chromaticity information.
Colour Mean and Variance	Acceptable; some noise on wing edges.	Robust to most illumination changes.	Acceptable; occasionally some holes.	OK	Normalised RGB; good
Colour and Gradient Fusion	Acceptable; edges detected.	Robust to most illumination changes.	Acceptable; more sensitive under low contrast.	Expensive	Slow.
GMM	Acceptable.	Robust to most illumination changes.	Acceptable; occasionally some holes.	OK	Number of Gaussians fixed. Large memory requirements. Slower to initialise.
KDE	Acceptable.	Robust to most illumination changes.	Acceptable.	Expensive.	Large memory requirements.
Linear Prediction	Acceptable.	Robust to most illumination changes.	Good sensitivity.	Very Expensive.	Prediction problems when pixels are misclassified as foreground.
RGB – Mean Value	Acceptable.	Robust to most illumination changes.	Acceptable	Linux: real time.	Issue of when to integrate objects into the background.

9. THE CHOSEN ALGORITHM(S)

At this stage of analysis, it was decided to eliminate those algorithms that do not satisfy the real-time constraint; mainly, the **Colour and Gradient Fusion** method, the **Kernel Density Estimation** method, and the **Linear Prediction** algorithm. When tested, these motion detectors were found to operate at a speed of 7.5, 5.3 and 2.7 frames per second respectively [2]. This is quite unfortunate for the Linear Prediction algorithm, as this motion detector appears to be the most sensitive in low contrast conditions (see frame 3300 in particular).

The **Chromatic Background Model** algorithm was also eliminated at this stage because of its low detection rate for objects with little colour information.

The remaining algorithms all appear to provide acceptable results on the tested video sequence. So it was decided to keep these algorithms at this stage of the AVITRACK project and to perform further testing as more video sequences become available with a wider range of environmental and weather conditions. For example, it may be that for night-time sequences a different one of these algorithms has to be selected, than the one used for day-time sequences.

The selected Motion Detection algorithms
Median and Morphology
Kalman Filter
Colour Mean and Variance
GMM
RGB – Mean Value

These algorithms will be integrated into the AVITRACK Scene Tracking module with the idea that one can switch from using one motion detector to another by just enabling or disabling an entry in the configuration file, without the need for re-compiling the code.

10. FUTURE WORK

- To obtain a better separation of foreground objects, a multi-layered approach to the background needs to be implemented. Foreground objects that become stationary need to be integrated into the background, with the ability of re-activating the objects once they start moving again. The main issue here is to determine how to create the background model layer for the stationary object, since very little history is available – especially for a motion detection algorithm using a complex background model. The object tracking algorithms can provide feedback on the stationary or non-stationary state of foreground objects. (Some of this functionality has already been implemented, but needs to be improved).
- Further testing with other data sets would be required to better understand the robustness and behaviour of the chosen motion detection algorithm to a wide range of environmental and weather conditions; for example, the effect of rain, fog on motion detection.
- In addition, the motion detector needs to be tested to night-time/dusk conditions where the colour information in the scene is expected to be minimal. For night-time conditions, model switching, or perhaps the use of a new algorithm, may be required.
- The effects of having direct sunlight in a camera's field-of-view, over-saturated images, reflections, etc., need to be examined with the possibility of using feedback from the results of the multiple cameras to validate the motion detection output.
- The elimination of strong dynamic shadows is also another issue that can be examined in the future.

11. REFERENCES

[1] PRIP Team, "Internal Technical Note – Motion Detection Report", IN_AVI_MOT_DET_PRIP.

[2] UoR Team, "Internal Technical Note – Motion Detection Report", IN_AVI_2_005.

[3] INRIA Team, "Internal Technical Note – Motion Detection Report".

[4] Ismail Haritaoglu, David Harwood and Larry S.Davis, "*W^4 Who? When? Where? What? A real time system for detecting and tracking people*", Apr 1998.

[5] M. Bramberger, R. Plugfelder, Arnold Maier, Berhard Rinner, Bernhard Strobl. *"A smart camera for traffic surveillance"*, In *Telecomunication and Mobile Computing Workshop in Graz (Austria)*, March 2003.

[6] K. Karmann, A. von Brandt "Moving Object Recognition Using an Adaptive Background Memory", in V. Cappellini (Ed.), *Time-varying Image Processing and Moving Object Recoginiton, 2*, pp. 289-296, Elsevier Publishers B.V. Amsterdam, The Netherlands, 1990.

[7] C. Ridder, O. Munkelt, H. Kirchner, “Adaptive Background Estimation and Foreground Detection using Kalman-Filtering” in O. Kaynak, M. Özkan, N. Bekiroglu, I. Tunay (Eds.) *International Conference on recdnt Advances in Mechatronics*, pp. 193-199, 1995.

[8] Thanarat Horprasert, David Harwood and Larry S.Davis. “A statistical approach for real-time robust background subtraction and shadow detection”. In Proc. IEEE ICCV FRAME-RATE Workshop, Kerkyra, Greece, September 1999.

[9] Nils T. Siebel, *“Design and Implementation of People Tracking Algorithms for Visual Surveillance Applications”*. PhD Thesis, The University of Reading, March 2003.

[10] Sumer Jabri, Zoran Duric, Harry Wechsler, Azriel Rosenfeld, *“Detection and Location of People in Video Images Using Adaptive Fusion of Color and Edge Information”*. In Proc. ICPR’00, 2000.

[11] Chris Stauffer, W.E.L. Grimson, *“Adaptive Background Mixture Models for Real-Time Tracking”*. In Proc. ICPR’99, 1999.

[12] Ahmed Elgammal, David Harwood, and Larry Davis, *“Non-parametric Model for Background Subtraction”*. In Proc. ICCV’99 FRAME-RATE Workshop, 1999.

[13] Kentaro Toyama, John Krumm, Barry Brumitt, and Brian Meyers, *“Wallflower: Principles and Practice of Background Maintenance”*. In Proc. ICCV’99, 1999.

[14] PRIP Team, “PRIP Meeting Minutes 01-02 July 2004”, AVI-PRIP-20040701-Meeting-Minutes.